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 abstract multi-context systems. However, the connection between model-based modular systems, on the other hand, and abstract multi-context systems also have a potential to offer insights into reasoning tools for hybrid modular knowledge representation formalisms.

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.

**Literal**s over a vocabulary \( \sigma \) are expressions \( a \) and \( \neg a \), where \( a \) ranges over \( \sigma \). A **logic program** over \( \sigma \) is a set of **rules** of the form

\[
a_0 \leftarrow a_1, \ldots, a_\ell, \text{ not } a_{\ell+1}, \ldots, \text{ not } a_m, \]

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

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 \( \text{hd}(\Pi) \) for the set of nonempty heads of rules in \( \Pi \). 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 \( \sigma \) is an **input answer set** of a logic program \( \Pi \) over \( \sigma \) if \( X \) is an answer set of the program \( \Pi \cup (X \setminus \text{hd}(\Pi)) \).

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 \( \mathcal{L} = (L, \sigma, \text{sem}) \), where

1. \( L \) is the set of propositional formulas over \( \sigma \),
2. \( \text{sem}(F) \), for a theory \( F \subseteq L \), is the set of propositional models of \( F \) over \( \sigma \),

captures propositional logic. We call this logic \( \mathcal{L} \) the **pl-logic** and modules in the **pl-logics**, **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 \( \mathcal{L} = (L, \sigma, \text{sem}) \), where

1. \( L \) is the set of logic program rules over \( \sigma \),
2. \( \text{sem}(\Pi) \), for a program \( \Pi \subseteq L \), is the set of input answer sets of \( \Pi \) over \( \sigma \).

We call this logic the \( \text{ilp-logic} \) and modules in this logic, \( \text{ilp-modules} \).

**Definition 5 (Lierler and Truszczynski 2013).** A modular logic program over a vocabulary \( \sigma \) is a set of logic programs over \( \sigma \). 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 \( \Pi_i \), \( 1 \leq i \leq n \), is a program over a vocabulary \( \sigma_i \). 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 \( \Pi_i \), \( 1 \leq i \leq n \), 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 ilp-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) (Marien 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 \( \sigma_{pl} \) and \( \Pi \) is a logic program over a vocabulary \( \sigma_{lp} \). 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 \( \Pi \).

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 \( \sigma_{pl} \) and \( \Pi \) is a logic program over a vocabulary \( \sigma_{lp} \). 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 \( \{F, \Pi\} \), where we treat \( F \) as a pl-module over \( \sigma_{pl} \) and \( \Pi \) as an ilp-module over \( \sigma_{lp} \).

**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 \( (t, \mathbb{R}) \), 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 \( \nu \) satisfies a constraint \((\{x_1, \ldots, x_n\}, \mathbb{R}) \) (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 \( c = (t, \mathbb{R}) \) be a constraint and \( D \) the domain of its variables. Let \( k \) denote the arity of \( t \). The constraint \( \tau = (t, D^k \setminus \mathbb{R}) \) is the complement (or dual) of \( c \). Clearly, an evaluation of variables in \( t \) satisfies \( c \) if and only if it does not satisfy \( \tau \).

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

For a CASP program \( P = \langle \Pi, B, \gamma \rangle \) over the vocabulary \( \sigma = \sigma_r \cup \sigma_c \) (understood as above), a set \( X \subseteq \sigma \) is an answer set of \( P \) if

\[
X \cap \sigma_r \subseteq hd(\Pi) \quad \text{and} \quad X \cap \sigma_c \subseteq At(\Pi)
\]

\[X \text{ is an input answer set of } \Pi, \text{ and}
\]
the following CSP has a solution

\[
\{ \gamma(a) : a \in X \cap \sigma_c \} \cup \{ \gamma(a) : a \in (At(\Pi) \cap \sigma_c) \setminus X \}.
\]

We will now introduce model-based modular systems designed to represent CASP programs in a class \( C \) of constraints.

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

\[
\{ c : c \in B \text{ and } c^n \in I \} \cup \{ \tau : c \in B \text{ and } c^n \notin I \}.
\]

We call the logic \( (C, \sigma_C, \text{sem}_C) \) a csp-logic determined by \( C \) and modules in this logic, csp-modules over \( C \).

Next, we define mapping modules. Let \( P = \langle \Pi, B, \gamma \rangle \) be a CASP program in a class \( C \) 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 \( \sigma_r \cup \sigma_c \) that is denoted by \( M_{\gamma} \) and defined by

\[M_{\gamma} = \{ a \mapsto (\gamma(a))^{c^n} : a \in At(\Pi) \cap \sigma_c \}.
\]

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 $\Pi$ over the vocabulary $\sigma_r \cup \sigma_c$, we define

$$\Pi^e = \Pi \cup \{\perp \leftarrow a : a \in \sigma_r \setminus hd(\Pi)\} \cup \{\perp \leftarrow a : a \in \sigma_c \setminus At(\Pi)\}. $$

Let $C$ be a class of constraints and $P = (\Pi, B, \gamma)$ a CAS program in $C$ over a signature $\sigma_r \cup \sigma_c$ of regular and constraint atoms. We define a modular system $P^m$ by setting $P^m = (\Pi^e, B, M_\gamma)$, where $\Pi^e$ is an ilp-module over the signature $\sigma_r \cup \sigma_c \cup \sigma_e$, $B \subseteq C$ is a csp-module over $C$, and $M_\gamma$ 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 $C$ be a class of constraints and $P = (\Pi, B, \gamma)$ a CAS program in $C$ 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 $X \cup \{(\gamma(a))^n : a \in X \cap \sigma_c\}$ is a model of the modular system $P^m$.

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

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 $P$, and (iii) the CSP

$$\{\gamma(a) : a \in X \cap \sigma_c\} \cup \{\gamma(a) : a \in (At(\Pi) \cap \sigma_e) \setminus X\}$$

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 $\{\perp \leftarrow a : a \in \sigma_r \setminus hd(\Pi)\} \cup \{\perp \leftarrow a : a \in \sigma_c \setminus At(\Pi)\}$. Thus, $X$ is an answer set of $\Pi \cup (X \setminus hd(\Pi))$.

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

It remains to show that $X \cup X_a^\gamma$ is a model of $B$ and $M_\gamma$.

To illustrate the former we have to show that the CSP

$$\{c : c \in B \text{ and } c^n \in X_a^\gamma\} \cup \{\tau : c \in B \text{ and } c^n \notin X_a^\gamma\}$$

has a solution. By the definition of CAS programs, $\gamma$ is a function from the set $At(\Pi) \cap \sigma_e$ to the set $B$. From the definition of $X_a^\gamma$, it follows that CSP (3) coincides with CSP (2).

By (iii), CSP (2) has a solution. The fact that $X \cup X_a^\gamma$ is a model of $M_\gamma$ follows immediately from the definition of $X_a^\gamma$.

Right-to-left: Let $X \cup X_a^\gamma$ be a model of $(\Pi^e, B, M_\gamma)$. It follows that $X \cup X_a^\gamma$ is an input answer set of $\Pi^e$. Reasoning as before, we obtain that $X$ is an input answer set of $\Pi^e$.

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 $\Pi$. The latter implies the inclusions $X \cap \sigma_r \subseteq hd(\Pi)$ and $X \cap \sigma_c \subseteq At(\Pi)$.

Since $X \cup X_a^\gamma$ is a model of $(\Pi^e, B, M_\gamma)$, it also follows that $X \cup X_a^\gamma$ is a model of $B$, and $X \cup X_a^\gamma$ is a model of $M_\gamma$.

These two properties imply that the CSP (2) has a solution using the same arguments as above. \qed

---

**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 $L_1, \ldots, L_n$ be logics, say $L_i = (L_i, \sigma_i, sem_i)$, $1 \leq i \leq n$, and let us assume that the vocabularies $\sigma_i$ of these logics are pairwise disjoint. A bridge rule over the logics $L_1, \ldots, L_n$ is an expression

$$b \leftarrow p_1, \ldots, p_k, \neg p_{k+1}, \ldots, \neg p_m,$$

where $b \in L_i$ for some logic $L_i$, $1 \leq i \leq n$, and each $p_j$, $1 \leq j \leq m$, is an element of the vocabulary $\sigma_k$ of some logic $L_k$ 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 $L_i$. For an interpretation $I$ of $\sigma$, we say that a rule $r$ is applicable in $I$ if $I$ satisfies the body of the rule $r$ identified with a propositional formula $p_1 \land \ldots \land p_k \land \neg p_{k+1} \land \ldots \land \neg p_m$, written $I \models bd(r)$.

We are now ready to define multi-context systems and their semantics. A multi-context system (MCS) over logics $L_1, \ldots, L_n$ (with pairwise disjoint vocabularies) is an $(n + 1)$-tuple $M = (M_1, \ldots, M_n, R)$, where for each $i$, $1 \leq i \leq n$, $M_i$ is an $L_i$-module, and $R$ is a set of bridge rules over $L_1, \ldots, L_n$. In the sequel, by $R_i$ we will denote the set of all bridge rules in $R$ that are into the logic $L_i$. An interpretation $I$ of $\sigma$ is an equilibrium model of $M$ if for every $i$, $1 \leq i \leq n$,

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

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

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 $\varphi$ is a conjunction of literals over $\sigma$, by a set of bridge rules \{$b \leftarrow \varphi : b \in B$\}. 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 $L_i = (L_i, \sigma_i, sem_i)$, $1 \leq i \leq n$, be logics with pairwise disjoint vocabularies, and let $\sigma = \sigma_1 \cup \ldots \cup \sigma_n$. We define logics $K_i = (K_i, \sigma, sem_i)$, $1 \leq i \leq n$, as follows. We set $K_i$ to consist of all expressions of the form $b \leftarrow \varphi$, where $b \in L_i$ and $\varphi$ is a conjunction of literals over $\sigma$. 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 $K_i$, $1 \leq i \leq n$, we specify the function $sem_i$. Namely, for a theory $T \subseteq K_i$, we define $sem_i(T)$ to contain an interpretation $I \in \text{Ird}(\sigma)$ if and only if $I_{\sigma_i} \models sem_i(\{hd(t) : t \in T, I \models bd(t)\})$.

Let $M = (M_1, \ldots, M_n, R)$ be a multi-context system over logics $L_1, \ldots, L_n$ 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 $K_i$-module and
(M_1',...,M_n') is a modular system over logics K_1,...,K_n. We denote this system by ms(H).

**Theorem 4.** Let H = (M_1,...,M_n,R) be a multi-context system over logics L_1,...,L_n with pairwise disjoint vocabularies \sigma_i. An interpretation I \in Int(\sigma), where \sigma = \sigma_1 \cup ... \cup \sigma_n, is an equilibrium model of H if and only if I is a model of the modular system ms(H).

**Proof.** Let I be an interpretation of \sigma. By the definition, I is an equilibrium model of H if and only if for every i, 1 \leq i \leq n,

\[ I_{\sigma_i} \in sem_i(M_i \cup \{hd(r): r \in R_i, r \text{ applicable in } I\}). \]

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

\[ M_i \cup \{hd(r): r \in R_i, r \text{ applicable in } I\} = M_i \cup \{hd(r): r \in R_i, I \models bd(r)\}. \]

Also by the definition, I is an equilibrium model of ms(H) if and only if for every i, 1 \leq i \leq n, I \in Int_\sigma(M_i'), that is,

\[ I_{\sigma_i} \in sem_i(\{hd(r): r \in M_i', I \models bd(r)\}). \]

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

\[ \{hd(r): r \in M_i', I \models bd(r)\} = M_i \cup \{hd(r): r \in R_i, I \models bd(r)\}. \]

Thus, the assertion follows. \square

**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 M = (M_1,...,M_n) over some logics L_i = (L_i,\sigma_i,sem_i), 1 \leq i \leq n, can be regarded as a modular system in logics L_i' = (L_i,\sigma,sem_i'), where

- \sigma = \sigma_1 \cup ... \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 L_i' are extensions of the logics 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 M = (M_1,...,M_n) be a modular system over logics L_i = (L_i,\sigma_i,sem_i), 1 \leq i \leq n. An interpretation X \in Int(\sigma) is a model of M if and only if I is a model of M viewed as a modular system over logics 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 M = (M_1,...,M_n) be a modular system over logics L_i = (L_i,\sigma_i,sem_i), 1 \leq i \leq n. We now construct a multi-context system ms(M), over different but closely related logics, that captures the semantics of M.

For i = 1,...,n and for every a \in \sigma, we introduce a fresh symbol a_i so that a_i \notin \sigma_1 \cup ... \cup \sigma_n. By \theta_i we denote a vocabulary \{a_i: a \in \sigma\}, whereas by \theta we denote a vocabulary \theta_1 \cup ... \cup \theta_n. For every i, 1 \leq i \leq n, we set L_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_i', by sem_i'(B) we denote the set that contains an interpretation I_i \in Int(\theta_i) precisely when the following two conditions hold:

1. \{a \in \sigma: a_i \in J_i\} \in sem_i(B \cap \theta_i).
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 N_i is a triple (L_i',\theta_i,sem_i'). It is clear that every L_i'-module B is also an N_i-module and its meaning in N_i is the same as in L_i (modulo the correspondence between the elements of the vocabularies \theta_i and \sigma). 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 L_i by multi-context systems over the logics N_i. For a modular system M = (M_1,...,M_n) over logics L_i = (L_i,\sigma_i,sem_i), we define the multi-context system ms(M) = (M_1,...,M_n,R), where every M_i is a module over the logic N_i, 1 \leq i \leq n, and R is a set of bridge rules R = \{a_i \leftarrow a_j: 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_i' of the logic N_i. We note that R depends only on n and on the common vocabulary \sigma of the logics L_1,...,L_n, and not on any particular choice of modules in a modular system over these logics.

**Theorem 6.** Let M = (M_1,...,M_n) be a modular system over logics L_i = (L_i,\sigma,sem_i). An interpretation I \in Int(\sigma) is a model of M if and only if \bigcup_{i=1}^n \{a_i: a \in I\} is an equilibrium model of the multi-context system ms(M).

Moreover, every equilibrium model of ms(M) is of such form.

**Proof.** For I \in Int(\sigma), we define I_i = \{a_i: a \in I\}, 1 \leq i \leq n, and \hat{I} = \bigcup_{i=1}^n I_i. Clearly, I_i \in Int(\theta_i) and \hat{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: a \in \sigma, i \neq j\}.

Therefore,

\[ \{hd(r): r \in R_i, r \text{ applicable in } \hat{I}\} = \{a_i: a \in I\}. \]

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

\[ I_i \in \hat{sem}_i(M_i \cup \{a_i: a \in I\}). \]

This amounts to showing that

\[ \{a \in \sigma: a_i \in I_i\} \in sem_i(M_i) \quad (5) \]
tions are straightforward, our discussion can be extended.

Right-to-left: Let \( J \in \text{Int}(\theta) \) be an equilibrium model of \( \text{mcs}(\mathcal{M}) \). By the definition, for every \( i, 1 \leq i \leq n, \)
\[
J[i] = \overline{\text{sem}}_{\mathcal{M}}(M_i \cup \{ \text{hd}(r) : r \in R_i, r \text{ is applicable in } J \}).
\]

By the condition (1) of the definition of \( \overline{\text{sem}}_{\mathcal{M}} \), we have \( \{ a \in \sigma : a_i \in J_i \} \in \text{sem}_{\mathcal{M}}(M_i) \). Moreover, let \( j \neq i \) and let \( a_j \in J \). The bridge rule \( a_i \xrightarrow{} a_j \) is applicable in \( J \) and so, we also have that \( a_i \in \{ \text{hd}(r) : r \in R_i, r \text{ is applicable in } J \} \).

By the condition (2) of the definition of \( \overline{\text{sem}}_{\mathcal{M}} \), we have \( a_i \in J_i \). 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 \( \hat{I} \) for some interpretation \( I \) of \( \sigma \) (this \( I \) can be explicitly defined as, for instance, \( I = \{ a \in \sigma : a_i \in J \} \)). Consequently, \( J[a_i] = I \) and so, \( \hat{I} \in \overline{\text{sem}}_{\mathcal{M}}(M_i) \). By the definition of \( \overline{\text{sem}}_{\mathcal{M}} \), \( I \in \text{sem}_{\mathcal{M}}(M_i) \). Thus, \( I \) is a model of the modular system \( \mathcal{M} \).

\[ \Box \]

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 \( \mathcal{L} = (L, \sigma_L, \text{sem}_L) \) be a logic, let \( \mathcal{B} \) be an \( L \)-module, and \( \psi \) be a vocabulary such that \( \psi \subseteq \sigma_L \). A pair \( (\mathcal{B}, \psi) \) is an mx-module over logic \( L \). An interpretation \( I \in \text{Int}(\sigma_L) \) is a model of an mx-module \( (\mathcal{B}, \psi) \) if \( I \) is a model of \( \mathcal{B} \). Vocabularies \( \psi \) and \( \sigma_L \setminus \psi \) 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 \( \mathcal{M}_1 = (B_1, \psi_1) \) and \( \mathcal{M}_2 = (B_2, \psi_2) \) be mx-modules over logics \( L_1 \) and \( L_2 \), respectively, and let \( \mathcal{M}_1 \triangleright \mathcal{M}_2 \) be defined. 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-modal systems (obtained by sequential composition) and model-based modular systems is evident.

**Proposition 7.** Let \( \mathcal{M}_1 = (B_1, \psi_1) \) and \( \mathcal{M}_2 = (B_2, \psi_2) \) be mx-modules over logics \( L_1 \) and \( L_2 \), respectively, and let \( \mathcal{M}_1 \triangleright \mathcal{M}_2 \) be defined. An interpretation \( I \) over \( \sigma_{\mathcal{M}_1} \cup \sigma_{\mathcal{M}_2} \), where \( \sigma_i \) is the vocabulary of the logic \( L_i, i = 1, 2 \), is a model of \( \mathcal{M}_1 \triangleright \mathcal{M}_2 \) if and only if \( I \) is a model of model-based modular system \( (B_1, B_2) \) over \( L_1 \) and \( L_2 \).

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-modal 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. 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.\footnote{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


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.
