1-2015

An Abstract View on Modularity in Knowledge Representation

Yuliya Lierler

Miroslaw Truszczynski

University of Kentucky

Follow this and additional works at: http://digitalcommons.unomaha.edu/compsicfacproc

Part of the Computer Sciences Commons

Recommended Citation

http://digitalcommons.unomaha.edu/compsicfacproc/40
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 $\sigma$ represent (classical propositional) interpretations of $\sigma$. We write $\text{Int}(\sigma)$ for the family of all interpretations of a vocabulary $\sigma$.

**Definition 1.** A logic is a triple $\mathcal{L} = (L, \sigma, \text{sem}_L)$, where

1. $L$ is a language (the language of the logic $\mathcal{L}$)
2. $\sigma$ is a vocabulary (the vocabulary of the logic $\mathcal{L}$)
3. $\text{sem}_L : 2^L \rightarrow 2^{\text{Int}(\sigma)}$ is a function assigning collections of interpretations to theories in $L$ (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 $\text{sem}$ maps theories into “belief sets” that are interpretations of some vocabulary $\sigma$ rather than elements from some arbitrary set.

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

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

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

**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 $I \in \text{Int}(\sigma_{\mathcal{H}})$ 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 $\sigma$ are expressions $a$ and $\neg a$, where $a$ ranges over $\sigma$. A logic program over $\sigma$ is a set of rules of the form

$$a_0 \leftarrow a_1, \ldots, a_k, \text{ not } a_{k+1}, \ldots, \text{ not } a_m, \quad (1)$$

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

1. $L$ is the set of logic program rules over $\sigma$, 

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

We call this logic the \textit{ilp-logic} and modules in this logic, \textit{ilp-modules}.

\textbf{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.

\textbf{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 \textit{ilp-module}.

It follows that abstract modular systems over the ilp-logic capture modular logic programs. Since modular logic programs subsume the formalism of pl-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, \textit{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.

\textbf{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 \( \sigma_F \) and \( \Pi \) is a logic program over a vocabulary \( \sigma_{pl} \). 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.

\textbf{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 \( \sigma_{lp} \).

\section*{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 \textit{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 \textit{regular} and \textit{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 \( (\Pi, B, \gamma) \), where \( \Pi \) is a logic program over the vocabulary \( \sigma \) 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 \( P = (\Pi, B, \gamma) \) over the vocabulary \( \sigma = \sigma_r \cup \sigma_c \) (understood as above), a set \( X \subseteq \sigma \) is an answer set\footnote{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.} of \( P \) if

\begin{itemize}
  \item \( X \cap \sigma_r \subseteq \text{hd}(\Pi) \) and \( X \cap \sigma_c \subseteq \text{At}(\Pi) \)
  \item \( X \) is an input answer set of \( \Pi \), and
  \item the following CSP has a solution
    
    \[ \{\gamma(a) : a \in X \cap \sigma_c\} \cup \{\overline{\gamma(a)} : a \in (\text{At}(\Pi) \cap \sigma_c) \setminus X\} \].
\end{itemize}

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

We start by defining a \textit{cs-logic} determined by \( C \). Theories of that cs-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 \( sem_C(B) \) to contain an interpretation \( I \) of \( \sigma_C \) precisely when the following CSP has a solution

\[ \{c : c \in B \text{ and } c^n \in I\} \cup \{\overline{v} : v \in B \text{ and } c^n \not\in I\} \].

We call the logic \( (C, \sigma_C, sem_C) \) a \textit{cs-logic} determined by \( C \) and modules in this logic, \textit{cs-modules over} \( C \).

Next, we define mapping modules. Let \( P = (\Pi, B, \gamma) \) be a CAS 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 \( \sigma_c \cup \sigma_r \) that is denoted by \( M_\gamma \) and defined by

\[ M_\gamma = \{a \leftrightarrow (\gamma(a))^n : a \in \text{At}(\Pi) \cap \sigma_c\} \].
Finally, for a logic program $\Pi$ over the vocabulary $\sigma_r \cup \sigma_c$, we define

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

Let $C$ be a class of constraints and $P = \langle \Pi, B, \gamma \rangle$ 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 $P^m = (\Pi^c, B, M_r)$, where $\Pi^c$ is an ilp-module over the signature $\sigma_r \cup \sigma_c \cup \sigma_c$, $B \subseteq C$ is a csp-module over $C$, and $M_r$ 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 $P = \langle \Pi, B, \gamma \rangle$ 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 $X \cap \sigma_r \subseteq \text{hd}(\Pi)$ and $X \cap \sigma_c \subseteq \text{At}(\Pi)$.

Proof. Let us set $X^c_n = \{ \gamma(a)^n: a \in X \cap \sigma_c \}$. Left-to-right: Let $X$ be an answer set of $P$. By the definition of an answer set, (i) $X \cap \sigma_r \subseteq \text{hd}(\Pi)$ and (ii) $X \cap \sigma_c \subseteq \text{At}(\Pi)$. From (i), it follows that $X$ satisfies all constraints in $\{ \bot \leftarrow a: a \in \sigma_r \setminus \text{hd}(\Pi) \} \cup \{ \bot \leftarrow a: a \in \sigma_c \setminus \text{At}(\Pi) \}$. Thus, $X$ is an answer set of

$$\Pi \cup \{ \bot \rightarrow a: a \in \sigma_r \setminus \text{hd}(\Pi) \} \cup \{ \bot \rightarrow a: a \in \sigma_c \setminus \text{At}(\Pi) \}.$$ 

Since $\text{hd}(\Pi^c) = \text{hd}(\Pi)$, $X$ is an answer set of $\Pi^c \cup \{ \bot \rightarrow a: a \in \sigma_r \setminus \text{hd}(\Pi) \}$. Thus, $X$ is an input answer set of $\Pi^c$ and, consequently, a model of the ilp-module $\Pi^c$. Since atoms $\gamma(a)^n$ are not in $\sigma_r \cup \sigma_c$, $X \cap X^c_n$ is a model of $\Pi^c$, too.

It remains to show that $X \cup X^c_n$ is a model of $B$ and $M_r$. To illustrate the former we have to show that the CSP

$$\{ c: c \in B \text{ and } c^n \in X^c_n \} \cup \{ \bar{c}: c \in B \text{ and } c^n \notin X^c_n \}$$

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

Right-to-left: Let $X \cup X^c_n$ be a model of $\langle \Pi^c, B, M_r \rangle$. It follows that $X \cup X^c_n$ is an input answer set of $\Pi^c$. Reasoning as before, we obtain that $X$ is an input answer set of $\Pi^c$.

Consequently, $X$ is an answer set of the program $\Pi \cup \{ \bot \rightarrow a: a \in \sigma_r \setminus \text{hd}(\Pi) \}$ and it satisfies all the constraints. The former implies that $X$ is an input set of $\Pi$. The latter implies the inclusions $X \cap \sigma_r \subseteq \text{hd}(\Pi)$ and $X \cap \sigma_c \subseteq \text{At}(\Pi)$.

Since $X \cup X^c_n$ is a model of $\langle \Pi^c, B, M_r \rangle$, it also follows that $X \cup X^c_n$ is a model of $B$, and $X \cup X^c_n$ is a model of $M_r$.

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 $L_1, \ldots, L_n$ be logics, say $L_i = (L_i, \sigma_i, \text{sem}_i)$, $1 \leq i \leq n$, 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

$$b \leftarrow p_1, \ldots, p_k, \text{not } p_{k+1}, \ldots, \text{not } p_m,$$

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 $\text{hd}(r)$ and its body by $\text{bd}(r)$. If the head of a bridge rule $r$ is an element of $L_i$, we say that the bridge rule is into $L_i$. For an interpretation $I$ of $\gamma$, 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 $I \models \text{bd}(r)$.

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 $\gamma$ is an equilibrium model of $M$ if for every $i$, $1 \leq i \leq n$,

$$I_{\sigma_i} \in \text{sem}_i(M_i \cup \{ \text{bd}(r): r \in R_i, r \text{ applicable in } I \}),$$

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 $\sigma_i$ 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 $L_i = (L_i, \sigma_i, \text{sem}_i)$, $1 \leq i \leq n$, be logics with pairwise disjoint vocabularies, and let $\sigma = \sigma_1 \cup \ldots \cup \sigma_n$. We define languages $K_i = (K_i, \sigma, \text{sem}_i)$, $1 \leq i \leq n$, 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 $\text{sem}_i$. Namely, for a theory $T \subseteq K_i$, we define $\text{sem}_i(T)$ to contain an interpretation $I \in \text{Int}(\sigma)$ if and only if $I_{|\sigma_i} \in \text{sem}_i(\{ \text{hd}(t): t \in T, I \models \text{bd}(t) \})$.

Let $H = (M_1, \ldots, M_n, R)$ 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
Thus, the assertion follows.

**Theorem 4.** Let \( \mathcal{H} = (M_1, \ldots, M_n, R) \) be a multi-context system over logics \( L_1, \ldots, L_n \) with pairwise disjoint vocabularies \( \sigma_i \). An interpretation \( I \in \text{Int}(\sigma) \), 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 \( \sigma \). By the definition, \( I \) is an equilibrium model of \( \mathcal{H} \) if and only if for every \( i, 1 \leq i \leq n, \)
\[
I[\sigma_i] \in \text{sem}_i(M_i \cup \{hd(r): r \in R_i, r \text{ applicable in } I\}).
\]
Since \( r \in R_i \) is applicable in \( I \) if and only if \( I[\sigma_i] = bd(r), \)
\[
M_i \cup \{hd(r): r \in R_i, r \text{ applicable in } I\}
= M_i \cup \{hd(r): r \in R_i, I[\sigma_i] = bd(r)\}.
\]
Also by the definition, \( I \) is a model of \( ms(\mathcal{H}) \) if and only if for every \( i, 1 \leq i \leq n, I \in \text{sem}_i(M_i) \), that is,
\[
I[\sigma_i] \in \text{sem}_i(M_r): r \in M_i, I[\sigma_i] = bd(r)\}.
\]
Since \( M_i = M_i \cup R_i \) and all elements in \( M_i \) have empty body,
\[
\{hd(r): r \in M_i, I[\sigma_i] = bd(r)\}
= M_i \cup \{hd(r): r \in R_i, I[\sigma_i] = bd(r)\}.
\]
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 \( L_i = (L_i, \sigma_i, \text{sem}_i), 1 \leq i \leq n \), can be regarded as a modular system in logics \( L_i' = (L_i, \sigma_i, \text{sem}_i') \), where
- \( \sigma = \sigma_1 \cup \ldots \cup \sigma_n \)
- for each \( i \) and each \( B \subseteq L_i, \text{sem}_i(B) \) contains an interpretation \( I \in \text{Int}(\sigma) \) if and only if \( I[\sigma_i] \in \text{sem}_i(B) \).

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 \( \mathcal{M} = (M_1, \ldots, M_n) \) be a modular system over logics \( L_i = (L_i, \sigma_i, \text{sem}_i), 1 \leq i \leq n \). An interpretation \( X \in \text{Int}(\sigma) \) is a model of \( \mathcal{M} \) if and only if \( I \) is a model of \( \mathcal{M} \) viewed as a modular system over logics \( L_i' = (L_i, \sigma, \text{sem}_i'), 1 \leq i \leq n \), where \( \text{sem}_i' \) 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 \( L_i = (L_i, \sigma_i, \text{sem}_i), 1 \leq i \leq n \). 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 \( \theta_i \), we denote a vocabulary \( \{a_i: a \in \sigma\} \), whereas by \( \theta \) we denote a vocabulary \( \theta_1 \cup \ldots \cup \theta_n \). For every \( i, 1 \leq i \leq n \), we set \( L_i^\theta = L_i \cup \theta_i \).

Since \( L_i \cap \theta_i = \emptyset \), there is never any ambiguity between elements of the two sets. For every \( B \subseteq L_i^\theta \), by \( \text{sem}_i(B) \) we denote the set that contains an interpretation \( I \in \text{Int}(\theta_i) \) precisely when the following two conditions hold:

1. \( \{a \in \sigma: a_i \in J_i\} \in \text{sem}_i(B \cap L) \).
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 \( (L_i^\theta, \theta_i, \text{sem}_i) \). It is clear that every \( L_i^\theta \)-module \( B \) is also an \( \mathcal{N}_i \)-module and its meaning in \( \mathcal{N}_i \) is precisely 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 \( \mathcal{N}_i \). For a modular system \( \mathcal{M} = (M_1, \ldots, M_n) \) over logics \( L_i = (L_i, \sigma_i, \text{sem}_i), 1 \leq i \leq n \), 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 \( \{a_i \leftarrow a_j: a_i \in \sigma, i \neq j\} \). These are indeed correctly formed bridge rules, as expressions \( a_i \) showing in the heads belong to the language \( L_i^\theta \) of the logic \( \mathcal{N}_i \). We note that \( R \) depends only on \( n \) and on the common vocabulary \( \sigma \) of the logics \( L_1, \ldots, L_n \), 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 \( L_i = (L_i, \sigma_i, \text{sem}_i) \). An interpretation \( I \in \text{Int}(\sigma) \) is a model of \( \mathcal{M} \) if and only if \( \bigcup_{i=1}^{n} \{a_i: a_i \in L_i\} \) 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 \( I \in \text{Int}(\sigma) \), we define \( I_i = \{a_i: a_i \in L_i\}, 1 \leq i \leq n \), and \( \tilde{I} = \bigcup_{i=1}^{n} I_i \). Clearly, \( I_i \in \text{Int}(\theta_i) \) and \( \tilde{I} \in \text{Int}(\theta) \).

- **Left-to-Right:** The set \( R_i \) of all bridge rules in \( R \) that are into a module \( M_i \) is given by \( R_i = \{a_i \leftarrow a_j: a_i \in \sigma, j \neq i\} \).
- **Right-to-Left:** Therefore,
\[
\{hd(r): r \in R_i, r \text{ applicable in } \tilde{I}\} = \{a_i: a_i \in I\}.
\]

Let \( I \in \text{Int}(\sigma) \) be a model of \( \mathcal{M} \). Since \( \tilde{I}_\theta = I_i \), to prove that \( \tilde{I} \) is an equilibrium model of \( mcs(\mathcal{M}) \), we need to show that for every \( i, 1 \leq i \leq n, \)
\[
I_i \in \text{sem}_i(M_i \cup \{a_i: a_i \in I\})
\]
This amounts to showing that
\[
\{a \in \sigma: a_i \in I_i\} \in \text{sem}_i(M_i)
\]
(the condition (2) of the definition of \( \text{sem}_i \) is evident). However, since \( I \) is a model of \( M \), \( I \) is a model of \( M_i \) in the logic \( \mathcal{L}_i \). Consequently, condition (5) holds.

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

By the condition (1) of the definition of \( \text{sem}_i \), we have \( \{ a \in \sigma : a_i \in J_{[i]} \} \in \text{sem}_i(M_i) \). 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 \( a_i \in \{ h_d(r) : r \in R_i, r \text{ is applicable in } J \} \).

By the condition (2) of the definition of \( \text{sem}_i \), 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, \( I = \{ a \in \sigma : a_i \in J_{[i]} \} \)). Consequently, \( J_{[i]} = I_i \) and so, \( I_i \in \text{sem}_i(M_i) \). By the definition of \( \text{sem}_i, I \in \text{sem}_i(M_i) \). Thus, \( I \) is a model of the modular system \( M \).

\[ \square \]

**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 (2011; 2013) introduced modular systems to the case when the 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 \( \mathcal{L} = (\mathcal{L}_i, \sigma_\mathcal{L}, \text{sem}_\mathcal{L}) \) be a logic, let \( \mathcal{B} \) be an \( \mathcal{L} \)-module, and \( \psi \) 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 \( I \in \text{Int}(\sigma_\mathcal{L}) \) is a model of an mx-module \((\mathcal{B}, \psi)\) if \( I \) is a model of \( \mathcal{B} \). Variables \( \psi \) 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 \( M_1 \) and \( M_2 \) are composable if output vocabularies of \( M_1 \) and \( M_2 \) are disjoint. An mx-module \( M_2 \) is independent from an mx-module \( M_1 \) if the input vocabulary of \( M_2 \) is disjoint from the output vocabulary of \( M_1 \). 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 \( \bowtie \) of sequential composition.

**Definition 8.** Let \( M_1 \) and \( M_2 \) be mx-modules in logics \( \sigma_1 \) and \( \sigma_2 \), and let \( M_1 \) and \( M_2 \) be composable and \( M_2 \) be independent from \( M_1 \). The expression \( M_1 \bowtie M_2 \) is an mx-system (constructed by sequential composition). An interpretation \( I \) over \( \sigma_{M_1} \cup \sigma_{M_2} \) is a model of \( M_1 \bowtie M_2 \) if \( I \mid \sigma_1 \) is a model of \( M_1 \) and \( I \mid \sigma_2 \) is a model of \( M_2 \).

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

**Proposition 7.** Let \( M_1 = (B_1, \psi_1) \) and \( M_2 = (B_2, \psi_2) \) be mx-modules over logics \( \mathcal{L}_1 \) and \( \mathcal{L}_2 \), respectively, and let \( M_1 \bowtie M_2 \) be defined. An interpretation \( I \) over \( \sigma_1 \cup \sigma_2 \), where \( \sigma_1 \) is the vocabulary of the logic \( \mathcal{L}_1 \), \( i = 1, 2 \), is a model of \( M_1 \bowtie M_2 \) if and only if \( I \) is a model of model-based modular system \((B_1, \psi_2)\) 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 compositability 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


