Module Theorem for The General Theory of Stable Models

Joseph Babb and Joohyung Lee

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA
(*e-mail:* {Joseph.Babb, joolee}@asu.edu)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

The module theorem by Janhunen *et al.* demonstrates how to provide a modular structure in answer set programming, where each module has a well-defined input/output interface which can be used to establish the compositionality of answer sets. The theorem is useful in the analysis of answer set programs, and is a basis of incremental grounding and reactive answer set programming. We extend the module theorem to the general theory of stable models by Ferraris *et al.* The generalization applies to non-ground logic programs allowing useful constructs in answer set programming, such as choice rules, the count aggregate, and nested expressions. Our extension is based on relating the module theorem to the symmetric splitting theorem by Ferraris *et al.* Based on this result, we reformulate and extend the theory of incremental answer set computation to a more general class of programs.

KEYWORDS: answer set programming, module theorem, splitting theorem

1 Introduction

The module theorem (Oikarinen and Janhunen 2008; Janhunen et al. 2009) demonstrates how to provide a modular structure for logic programs under the stable model semantics, where each module has a well-defined input/output interface which can be used to establish the compositionality of answer sets of different modules. The theorem was shown to be useful in the analysis of answer set programs and was used as a basis of incremental grounding (Gebser et al. 2008) and reactive answer set programming (Gebser et al. 2011), resulting in systems ICLINGO and OCLINGO.

The module theorem was stated for normal logic programs and SMODELS programs in (Oikarinen and Janhunen 2008) and for disjunctive logic programs in (Janhunen et al. 2009), but both papers considered ground programs only. In this paper we extend the module theorem to non-ground programs, or more generally, to first-order formulas under the stable model semantics proposed by Ferraris et al. (2011). We derive the generalization by relating the module theorem to the symmetric splitting theorem by Ferraris et al. (2009). This is expected in some sense as the

symmetric splitting theorem looks close to the module theorem and is already applicable to first-order formulas under the stable model semantics (Ferraris et al. 2011). Since non-ground logic programs can be understood as a special class of first-order formulas under the stable model semantics, the theorem can be applied to split these programs. In addition, as the semantics of choice rules and the count aggregate in answer set programming is understood as shorthand for some first-order formulas (Lee et al. 2008), the splitting theorem can also be applied to non-ground programs containing such constructs.

The precise relationship between the module theorem and the splitting theorem has not been established, partly because there is some technical gap that needs to be closed. While the splitting theorem is applicable to more general classes of programs in most cases, there are some cases where the module theorem allows us to split, but the splitting theorem does not.

In order to handle this issue, we first extend the splitting theorem to allow this kind of generality. We then add modular structures to the splitting theorem, and provide a mechanism of composing partial interpretations for each module. This new theorem serves as the module theorem for the general theory of stable models.

The paper is organized as follows. In the next section, we review the stable model semantics from (Ferraris et al. 2011), the splitting theorem, and the module theorem. In Section 3 we provide a generalization of the splitting theorem, which closes the gap between the module theorem and the splitting theorem. In Section 4 we present the module theorem for the general theory of stable models, which extends both the previous splitting theorem and the previous module theorem. We give an example of the generalized module theorem in Section 5 and show how it serves as a foundation for extending the theory of incremental answer set computation in Section 6.

2 Preliminaries

2.1 Review: General Theory of Stable Models

This review follows the definition by Ferraris et al. (2011). There, stable models are defined in terms of the SM operator, which is similar to the circumscription operator CIRC (Lifschitz 1994).

Let $\mathbf{p}$ be a list of distinct predicate constants $p_1, \dots, p_n$, and let $\mathbf{u}$ be a list of distinct predicate variables $u_1, \dots, u_n$. By $\mathbf{u} \leq \mathbf{p}$ we denote the conjunction of the formulas $\forall \mathbf{x}(u_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))$ for all $i = 1, \dots, n$, where $\mathbf{x}$ is a list of distinct object variables whose length is the same as the arity of p_i. Expression $\mathbf{u} < \mathbf{p}$ stands for $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{p} \leq \mathbf{u})$. For instance, if p and q are unary predicate constants then $(u, v) < (p, q)$ is

$$\forall x(u(x) \rightarrow p(x)) \wedge \forall x(v(x) \rightarrow q(x)) \wedge \neg\Big(\forall x(p(x) \rightarrow u(x)) \wedge \forall x(q(x) \rightarrow v(x))\Big).$$

For any first-order formula F, $\mathrm{SM}[F; \mathbf{p}]$ is defined as

$$F \wedge \neg\exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u})),$$

where $F^*(\mathbf{u})$ is defined recursively as follows: [1]

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ for any list $\mathbf{t}$ of terms;
- $F^* = F$ for any atomic formula F (including $\bot$ and equality) that does not contain members of $\mathbf{p}$;
- $(F \wedge G)^* = F^* \wedge G^*$;
- $(F \vee G)^* = F^* \vee G^*$;
- $(F \to G)^* = (F^* \to G^*) \wedge (F \to G)$;
- $(\forall x F)^* = \forall x F^*$;
- $(\exists x F)^* = \exists x F^*$.

When F is a sentence, the models of $\mathrm{SM}[F;\mathbf{p}]$ are called the $\mathbf{p}$-*stable* models of F. Intuitively, they are the models of F that are "stable" on $\mathbf{p}$. We will often simply write $\mathrm{SM}[F]$ in place of $\mathrm{SM}[F;\mathbf{p}]$ when $\mathbf{p}$ is the list of all predicate constants occurring in F, and often identify $\mathbf{p}$ with the corresponding set if there is no confusion.

By an *answer set* of F that contains at least one object constant we understand an Herbrand interpretation of $\sigma(F)$ that satisfies $\mathrm{SM}[F]$, where $\sigma(F)$ is the signature consisting of the object, function and predicate constants occurring in F.

The answer sets of a logic program Π are defined as the answer sets of the FOL-representation of Π (i.e., the conjunction of the universal closures of implications corresponding to the rules). For example, the FOL-representation F of the program

$$\begin{array}{l} p(a) \\ q(b) \\ r(x) \leftarrow p(x), \mathit{not}\ q(x) \end{array}$$

is

$$p(a) \wedge q(b) \wedge \forall x(p(x) \wedge \neg q(x) \to r(x)) \tag{1}$$

and $\mathrm{SM}[F]$ is

$$\begin{array}{l} p(a) \wedge q(b) \wedge \forall x(p(x) \wedge \neg q(x) \to r(x)) \\ \quad \wedge \neg \exists uvw(((u,v,w) < (p,q,r)) \wedge u(a) \wedge v(b) \\ \qquad \wedge \forall x((u(x) \wedge (\neg v(x) \wedge \neg q(x)) \to w(x)) \wedge (p(x) \wedge \neg q(x) \to r(x)))), \end{array}$$

which is equivalent to the first-order sentence

$$\forall x(p(x) \leftrightarrow x = a) \wedge \forall x(q(x) \leftrightarrow x = b) \wedge \forall x(r(x) \leftrightarrow (p(x) \wedge \neg q(x))) \tag{2}$$

(Ferraris et al. 2007, Example 3). The stable models of F are any first-order models of (2). The only answer set of F is the Herbrand model $\{p(a),\ q(b),\ r(a)\}$.

Ferraris et al. (2011) show that this definition of an answer set, when applied to the syntax of logic programs, is equivalent to the traditional definition of an answer set that is based on grounding and fixpoints (Gelfond and Lifschitz 1988).

[1] We understand $\neg F$ as shorthand for $F \to \bot$.

2.2 Review: Symmetric Splitting Theorem

We say that an occurrence of a predicate constant, or any other subexpression, in a formula F is *positive* if the number of implications containing that occurrence in the antecedent is even (recall that we treat $\neg G$ as shorthand for $G \to \bot$). We say that the occurrence is *strictly positive* if the number of implications in F containing that occurrence in the antecedent is 0. For example, in (1), both occurrences of q are positive, but only the first one is strictly positive. A *rule* of F is an implication that occurs strictly positively in F.

A formula F is called *negative* on a list $\mathbf{p}$ of predicate constants if members of $\mathbf{p}$ have no strictly positive occurrences in F. For example, formula (1) is negative on $\{s\}$, but is not negative on $\{p, q\}$. A formula of the form $\neg F$ (shorthand for $F \to \bot$) is negative on any list of predicate constants.

The following definition of a dependency graph is from (Lee and Palla 2012), which is similar to the one from (Ferraris et al. 2009), but may contain less edges.

Definition 1 (*Predicate Dependency Graph*)
The *predicate dependency graph* of a first-order formula F relative to $\mathbf{p}$, denoted by $\mathrm{DG}[F; \mathbf{p}]$, is the directed graph that

- has all members of $\mathbf{p}$ as its vertices, and
- has an edge from p to q if, for some rule $G \to H$ of F,
  - — p has a strictly positive occurrence in H, and
  - — q has a positive occurrence in G that does not belong to any subformula of G that is negative on $\mathbf{p}$.

For example, $\mathrm{DG}[(1);\ p, q, r]$ has the vertices p, q, and r, and a single edge from r to p.

Theorem 1 (*Splitting Theorem, (Ferraris et al. 2009)*)
Let F, G be first-order sentences, and let $\mathbf{p}$, $\mathbf{q}$ be finite disjoint lists of distinct predicate constants. If

(a) each strongly connected component of the predicate dependency graph of $F \wedge G$ relative to $\mathbf{p}$, $\mathbf{q}$ is a subset of $\mathbf{p}$ or a subset of $\mathbf{q}$,
(b) F is negative on $\mathbf{q}$, and
(c) G is negative on $\mathbf{p}$

then

$$\mathrm{SM}[F \wedge G;\ \mathbf{pq}] \leftrightarrow \mathrm{SM}[F;\ \mathbf{p}] \wedge \mathrm{SM}[G;\ \mathbf{q}]$$

is logically valid.

Theorem 1 is slightly more generally applicable than the version of the splitting theorem from (Ferraris et al. 2009) as it refers to the refined definition of a dependency graph above instead of the one considered in (Ferraris et al. 2009).

Example 1
Theorem 1 tells us that SM[(1)] is equivalent to

$$\mathrm{SM}[p(a) \wedge q(b);\ p, q] \wedge \mathrm{SM}[\forall x(p(x) \wedge \neg q(x) \to r(x));\ r].$$

2.3 Review: DLP-Modules and Module Theorem

Janhunen et al. (2009) considered rules of the form

$$a_1; \ldots; a_n \leftarrow b_1, \ldots, b_m, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_k \tag{3}$$

where $n, m, k \geq 0$ and $a_1, \ldots, a_n, b_1, \ldots b_m, c_1, \ldots, c_k$ are propositional atoms. They define a *DLP-module* as a quadruple $(\Pi, \mathcal{I}, \mathcal{O}, \mathcal{H})$, where Π is a finite propositional disjunctive logic program consisting of rules of the form (3), and $\mathcal{I}$, $\mathcal{O}$, and $\mathcal{H}$ are finite sets of propositional atoms denoting the input, output, and hidden atoms, respectively, such that (i) the sets of input, output, and hidden atoms are disjoint; (ii) every atom occurring in Π is either an input, output, or hidden atom; (iii) every rule in Π with a nonempty head contains at least one output or hidden atom.

A module's hidden atoms can be viewed as a special case of its output atoms which occur in no other modules. For simplicity, we consider only DLP-modules with no hidden atoms ($\mathcal{H} = \emptyset$), which we denote by a triple $(\Pi, \mathcal{I}, \mathcal{O})$.

Definition 2 (*Module Answer Set, (Janhunen et al. 2009)*)
We say that a set X of atoms is a *(module) answer set* of a DLP-module $(\Pi, \mathcal{I}, \mathcal{O})$ if X is an answer set of $\Pi \cup \{p \mid p \in (\mathcal{I} \cap X)\}$.

The role of input atoms can be simulated using choice rules. A choice rule $\{p\} \leftarrow \mathit{Body}$ is understood as shorthand for $p; \mathit{not}\ p \leftarrow \mathit{Body}$ (Lee et al. 2008). The following lemma shows how module answer sets can be alternatively characterized in terms of choice rules.

Lemma 1
X is a module answer set of $(\Pi, \mathcal{I}, \mathcal{O})$ iff X is an answer set of $\Pi \cup \{\{p\} \leftarrow \mid p \in \mathcal{I}\}$.

Definition 3 (*Dependency Graph of a DLP-Module*)
The *dependency graph* of a DLP-module $\mathbf{\Pi} = (\Pi, \mathcal{I}, \mathcal{O})$, denoted by $\mathrm{DG}[\Pi; \mathcal{O}]$, is the directed graph that

- has all members of $\mathcal{O}$ as its vertices, and
- has edges from each a_i $(1 \leq i \leq n)$ to each b_j $(1 \leq j \leq m)$ for each rule (3) in Π.

It is clear that this definition is a special case of Definition 1.

Definition 4 (*Joinability of DLP-modules*)
Two DLP-modules $\mathbf{\Pi_1} = (\Pi_1, \mathcal{I}_1, \mathcal{O}_1)$ and $\mathbf{\Pi_2} = (\Pi_2, \mathcal{I}_2, \mathcal{O}_2)$ are called *joinable* if

- $\mathcal{O}_1 \cap \mathcal{O}_2 = \emptyset$,
- each strongly connected component of $\mathrm{DG}[\Pi_1 \cup \Pi_2;\ \mathcal{O}_1\mathcal{O}_2]$ is either a subset of $\mathcal{O}_1$ or a subset of $\mathcal{O}_2$,
- each rule in Π_1 (Π_2, respectively) whose head is not disjoint with $\mathcal{O}_2$ ($\mathcal{O}_1$, respectively) occurs in Π_2 (Π_1, respectively).

Definition 5 (*Join of DLP-modules*)
For any modules $\mathbf{\Pi_1} = (\Pi_1, \mathcal{I}_1, \mathcal{O}_1)$ and $\mathbf{\Pi_2} = (\Pi_2, \mathcal{I}_2, \mathcal{O}_2)$ that are joinable, the *join* of $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$, denoted by $\mathbf{\Pi}_1 \sqcup \mathbf{\Pi}_2$, is defined to be the DLP-module

$$(\Pi_1 \cup \Pi_2,\ (\mathcal{I}_1 \cup \mathcal{I}_2) \setminus (\mathcal{O}_1 \cup \mathcal{O}_2),\ \mathcal{O}_1 \cup \mathcal{O}_2)\ .$$

Informally, the join of two DLP-modules corresponds to the union of their programs, and defines all atoms that are defined by either module.

Given sets of atoms X_1, X_2, and A, we say that X_1 and X_2 are *A-compatible* if $X_1 \cap A = X_2 \cap A$. As demonstrated by Janhunen et al. (2009), given a program composed of a series of joinable DLP-modules, it is possible to consider each DLP-module contained in a program separately, evaluate them, and compose the resulting compatible answer sets in order to obtain the answer sets of the complete program. This notion is presented in Theorem 2, which is a reformulation of the main theorem (Theorem 5.7) from (Janhunen et al. 2009).

Theorem 2 (*Module Theorem for DLPs*)
Let $\mathbf{\Pi}_1 = (\Pi_1, \mathcal{I}_1, \mathcal{O}_1)$ and $\mathbf{\Pi}_2 = (\Pi_2, \mathcal{I}_2, \mathcal{O}_2)$ be DLP-modules that are joinable, and let X_1 and X_2 be $((\mathcal{I}_1 \cup \mathcal{O}_1) \cap (\mathcal{I}_2 \cup \mathcal{O}_2))$-compatible sets of atoms. The set $X_1 \cup X_2$ is a module answer set of $\mathbf{\Pi}_1 \sqcup \mathbf{\Pi}_2$ iff X_1 is a module answer set of $\mathbf{\Pi}_1$ and X_2 is a module answer set of $\mathbf{\Pi}_2$.

3 A Generalization of the Splitting Theorem by Ferraris *et al.*

The module theorem (Theorem 2) and the splitting theorem (Theorem 1) resemble each other. When we restrict attention to propositional logic program F, the intensional predicates $\mathbf{p}$ in $\mathrm{SM}[F; \mathbf{p}]$ correspond to output atoms in the corresponding module. Though not explicit in the notation $\mathrm{SM}[F; \mathbf{p}]$, the predicates that are not in $\mathbf{p}$ behave like input atoms in the corresponding module. Also, the joinability condition in Definition 4 appears similar to the splitting condition in Theorem 1, but with one exception: the last clause in the definition of joinability (Definition 4) does not have a counterpart in the splitting theorem. The module theorem allows us to join two DLP-modules $\mathbf{\Pi}_1 = (\Pi_1, \mathcal{I}_1, \mathcal{O}_1)$ and $\mathbf{\Pi}_2 = (\Pi_2, \mathcal{I}_2, \mathcal{O}_2)$ even when Π_1 has a rule whose head contains an output atom in $\mathcal{O}_2$ as long as that rule is also in Π_2.

Indeed, this difference yields the splitting theorem less generally applicable than the module theorem in some cases. For example, the module theorem (Theorem 2) allows us to join

$$\begin{aligned} \mathbf{\Pi}_1 &= (\{p \vee q \leftarrow r. \quad s \leftarrow .\},\ \{q, r\},\ \{p, s\}) \text{ and} \\ \mathbf{\Pi}_2 &= (\{p \vee q \leftarrow r. \quad t \leftarrow .\},\ \{p, r\},\ \{q, t\}) \end{aligned} \tag{4}$$

into

$$\mathbf{\Pi} = (\{p \vee q \leftarrow r. \quad s \leftarrow .\ \ t \leftarrow .\},\ \{r\},\ \{p, q, s, t\}).$$

On the other hand, the splitting theorem (Theorem 1), as presented in Section 2.2,

is not as general in this regard. It does not allow us to justify that

$$\mathrm{SM}[(r \to p \vee q) \wedge s;\ p, s] \wedge \mathrm{SM}[(r \to p \vee q) \wedge t;\ q, t] \tag{5}$$

is equivalent to

$$\mathrm{SM}[(r \to p \vee q) \wedge s \wedge t;\ p, q, s, t]\ . \tag{6}$$

because, for instance, $r \to p \vee q$ in the first conjunctive term of (5) is not negative on $\{q, t\}$.

In order to close the gap, we next extend the splitting theorem to allow a *partial split*, which allows an overlapping sentence, such as $r \to p \vee q$ in the above example, in both component formulas.

Theorem 3 (*Extension of the Splitting Theorem*)
Let F, G, H be first-order sentences, and let $\mathbf{p}$, $\mathbf{q}$ be finite lists of distinct predicate constants. If

(a) each strongly connected component of $\mathrm{DG}[F \wedge G \wedge H;\ \mathbf{pq}]$ is a subset of $\mathbf{p}$ or a subset of $\mathbf{q}$,
(b) F is negative on $\mathbf{q}$, and
(c) G is negative on $\mathbf{p}$

then

$$\mathrm{SM}[F \wedge G \wedge H;\ \mathbf{pq}] \leftrightarrow \mathrm{SM}[F \wedge H;\ \mathbf{p}] \wedge \mathrm{SM}[G \wedge H;\ \mathbf{q}]$$

is logically valid.

It is clear that Theorem 1 is a special case of Theorem 3 (take H to be $\top$). Unlike in (Ferraris et al. 2009) we do not require $\mathbf{p}$ and $\mathbf{q}$ to be disjoint from each other.

Getting back to the example above, according to the extended splitting theorem, (6) is equivalent to (5) (Take H to be $r \to p \vee q$).

4 Module Theorem for General Theory of Stable Models

4.1 Statement of the Theorem

In this section, we present a new formulation of the module theorem that is applicable to first-order formulas under the stable model semantics.

As a step towards this end, we first define the notion of a partial interpretation. Given a signature σ and its subset $\mathbf{c}$, by a $\mathbf{c}$-*partial interpretation* of σ, we mean an interpretation of σ restricted to $\mathbf{c}$. Clearly, a σ-partial interpretation of σ is simply an interpretation of σ. By an *Herbrand* $\mathbf{c}$-partial interpretation of σ, we mean an Herbrand interpretation of σ restricted to $\mathbf{c}$.

We say that a $\mathbf{c}_1$-partial interpretation I_1 and a $\mathbf{c}_2$-partial interpretation I_2 of the same signature σ are *compatible* if their universes are the same, and $c^{I_1} = c^{I_2}$ for every common constant c in $\mathbf{c}_1 \cap \mathbf{c}_2$. For such compatible partial interpretations I_1 and I_2, we define the *union* of I_1 and I_2, denoted by $I_1 \cup I_2$, to be the $(\mathbf{c_1} \cup \mathbf{c_2})$-partial interpretation of σ such that (i) $|I_1 \cup I_2| = |I_1| = |I_2|$ [2], (ii) $c^{I_1 \cup I_2} = c^{I_1}$ for every constant c in $\mathbf{c}_1$, and (iii) $c^{I_1 \cup I_2} = c^{I_2}$ for every constant c in $\mathbf{c}_2$.

[2] $|I|$ denotes the universe of the interpretation I.

Next we introduce a first-order analog to DLP-modules, which we refer to as *first-order modules*, and define a method of composing multiple such constructs similar to the join operation for DLP-modules. By $pr(F)$ we denote the set of all predicate constants occurring in F. A *(first-order) module* $\mathbf{F}$ of a signature σ is a triple $(F, \mathcal{I}, \mathcal{O})$, where F is a first-order sentence of σ, and $\mathcal{I}$ and $\mathcal{O}$ are disjoint lists of distinct predicate constants of σ such that $pr(F) \subseteq (\mathcal{I} \cup \mathcal{O})$. Intuitively, $\mathcal{I}$ and $\mathcal{O}$ denote, respectively, the sets of non-intensional (input) and intensional (output) predicates considered by F.

Definition 6 (*Module Stable Model*)
We say that an interpretation I is a *(module) stable model* of a module $\mathbf{F} = (F, \mathcal{I}, \mathcal{O})$ if $I \models \mathrm{SM}[F; \mathcal{O}]$. We understand $\mathrm{SM}[\mathbf{F}]$ as shorthand for $\mathrm{SM}[F; \mathcal{O}]$.

Definition 7 (*Joinability of First-Order Modules*)
Two first-order modules $\mathbf{F}_1 = (F_1 \wedge H,\ \mathcal{I}_1,\ \mathcal{O}_1)$ and $\mathbf{F}_2 = (F_2 \wedge H,\ \mathcal{I}_2,\ \mathcal{O}_2)$ are called *joinable* if

- $\mathcal{O}_1 \cap \mathcal{O}_2 = \emptyset$,
- each strongly connected component of $\mathrm{DG}[F_1 \wedge F_2 \wedge H;\ \mathcal{O}_1 \cup \mathcal{O}_2]$ is either a subset of $\mathcal{O}_1$ or a subset of $\mathcal{O}_2$,
- F_1 is negative on $\mathcal{O}_2$, and
- F_2 is negative on $\mathcal{O}_1$.

Definition 8 (*Join of First-Order modules*)
For any modules $\mathbf{F}_1 = (F_1 \wedge H,\ \mathcal{I}_1,\ \mathcal{O}_1)$ and $\mathbf{F}_2 = (F_2 \wedge H,\ \mathcal{I}_2,\ \mathcal{O}_2)$ that are joinable, the *join* of $\mathbf{F}_1$ and $\mathbf{F}_2$, denoted by $\mathbf{F}_1 \sqcup \mathbf{F}_2$, is defined to be the first-order module

$$(F_1 \wedge F_2 \wedge H,\ (\mathcal{I}_1 \cup \mathcal{I}_2) \setminus (\mathcal{O}_1 \cup \mathcal{O}_2),\ \mathcal{O}_1 \cup \mathcal{O}_2)\ .$$

It is not difficult to check that this definition is a proper generalization of Definition 4.

As with DLP-modules, the join operation for first-order modules is both commutative and associative.

Proposition 1 (*Commutativity and Associativity of Join*)
For any first-order modules $\mathbf{F}_1$, $\mathbf{F}_2$, and $\mathbf{F}_3$, the following properties hold:

- $\mathbf{F}_1 \sqcup \mathbf{F}_2$ is defined iff $\mathbf{F}_2 \sqcup \mathbf{F}_1$ is defined.
- $\mathrm{SM}[\mathbf{F}_1 \sqcup \mathbf{F}_2]$ is equivalent to $\mathrm{SM}[\mathbf{F}_2 \sqcup \mathbf{F}_1]$.
- $(\mathbf{F}_1 \sqcup \mathbf{F}_2) \sqcup \mathbf{F}_3$ is defined iff $\mathbf{F}_1 \sqcup (\mathbf{F}_2 \sqcup \mathbf{F}_3)$ is defined.
- $\mathrm{SM}[(\mathbf{F}_1 \sqcup \mathbf{F}_2) \sqcup \mathbf{F}_3]$ is equivalent to $\mathrm{SM}[\mathbf{F}_1 \sqcup (\mathbf{F}_2 \sqcup \mathbf{F}_3)]$.

The following theorem is an extension of Theorem 2 to the general theory of stable models. Given a first-order formula F, by $\mathbf{c}(F)$ we denote the set of all object, function and predicate constants occurring in F.

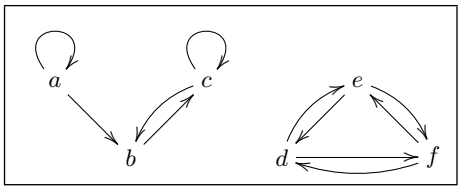

Fig. 1. A Simple Graph

Theorem 4 (*Module Theorem for General Theory of Stable Models*)

Let $\mathbf{F}_1 = (F_1, \mathcal{I}_1, \mathcal{O}_1)$ and $\mathbf{F}_2 = (F_2, \mathcal{I}_2, \mathcal{O}_2)$ be first-order modules of a signature σ that are joinable, and, for $i = 1, 2$, let $\mathbf{c}_i$ be a subset of σ that contains $\mathbf{c}(F_i) \cup \mathcal{O}_i$, and let I_i be a $\mathbf{c}_i$-partial interpretation of σ. If I_1 and I_2 are compatible with each other, then

$$I_1 \cup I_2 \models \mathrm{SM}[\mathbf{F}_1 \sqcup \mathbf{F}_2] \qquad \text{iff} \qquad I_1 \models \mathrm{SM}[\mathbf{F}_1] \ \text{ and } \ I_2 \models \mathrm{SM}[\mathbf{F}_2] \ .$$

It is clear that when $\sigma = \mathbf{c}_1 = \mathbf{c}_2$, Theorem 4 reduces to Theorem 3.

Also, it is not difficult to check that Theorem 4 reduces to Theorem 2 when $\mathbf{F}_1$ and $\mathbf{F}_2$ represent DLP-modules, $\mathbf{c}_1$ is $\mathcal{I}_1 \cup \mathcal{O}_1$, and $\mathbf{c}_2$ is $\mathcal{I}_2 \cup \mathcal{O}_2$.

5 Example: Analyzing RASPL-1 Programs Using Module Theorem

As an example of Theorem 4, consider the problem of locating non-singleton cliques within a graph, such as the one shown in Figure 1, that are reachable from a pre-specified node. This problem can be divided into three essential parts: (*i*) fixing the graph, (*ii*) determining the reachable subgraph, and (*iii*) locating cliques within that subgraph.

We can describe the graph shown in Figure 1 in the language of RASPL-1 (Lee et al. 2008), which is essentially a fragment of the general theory of stable models in logic programming syntax. We assume that σ is an underling signature. The program below lists the vertices and the edges using predicates `vertex` and `edge`, and assigns the starting vertex using `at` predicate.

$$\begin{array}{llllll} \texttt{vertex}(a). & \texttt{vertex}(b). & \texttt{vertex}(c). & \texttt{vertex}(d). & \texttt{vertex}(e). & \texttt{vertex}(f). \\ \texttt{edge}(a,a). & \texttt{edge}(a,b). & \texttt{edge}(b,c). & \texttt{edge}(c,b). & \texttt{edge}(c,c). & \texttt{edge}(d,e). \\ \texttt{edge}(d,f). & \texttt{edge}(e,d). & \texttt{edge}(e,f). & \texttt{edge}(f,d). & \texttt{edge}(f,e). & \texttt{at}(a). \end{array} \tag{7}$$

The first-order module $\mathbf{F}_G$ is $(F_G, \emptyset, \{\texttt{vertex}, \texttt{edge}, \texttt{at}\})$, where F_G is the FOL-representation of program (7), which is the conjunction of all the atoms. Let I_G be the following Herbrand $\mathbf{c}(F_G)$-partial interpretation of σ that satisfies $\mathrm{SM}[\mathbf{F}_G]$.

$$\begin{aligned}
\texttt{vertex}^{I_G} &= \{\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{e},\texttt{f}\},\\
\texttt{edge}^{I_G} &= \{(\texttt{a},\texttt{a}),(\texttt{a},\texttt{b}),(\texttt{b},\texttt{c}),(\texttt{c},\texttt{b}),\\
&\qquad (\texttt{c},\texttt{c}),(\texttt{d},\texttt{e}),(\texttt{d},\texttt{f}),(\texttt{e},\texttt{d}),(\texttt{e},\texttt{f}),(\texttt{f},\texttt{d}),(\texttt{f},\texttt{e})\},\text{ and}\\
\texttt{at}^{I_G} &= \{\texttt{a}\}.
\end{aligned}$$

The following program describes the reachable vertices by the predicate `reachable`, which is defined using `edge` and `at`.

$$\begin{aligned}
&\texttt{reachable}(X) \leftarrow \texttt{at}(X).\\
&\texttt{reachable}(Y) \leftarrow \texttt{reachable}(X), \texttt{edge}(X,Y).
\end{aligned} \tag{8}$$

The first-order module $\mathbf{F}_R$ is $(F_R, \{\texttt{edge},\texttt{at}\}, \{\texttt{reachable}\})$, where F_R is the FOL-representation of program (8). Let I_R be the following Herbrand $\mathbf{c}(F_R)$-partial interpretation of σ that satisfies SM$[\mathbf{F}_G]$, which is compatible with I_G.

$$\begin{aligned}
\texttt{edge}^{I_R} &= \{(\texttt{a},\texttt{a}),(\texttt{a},\texttt{b}),(\texttt{b},\texttt{c}),(\texttt{c},\texttt{b}),\\
&\qquad (\texttt{c},\texttt{c}),(\texttt{d},\texttt{e}),(\texttt{d},\texttt{f}),(\texttt{e},\texttt{d}),(\texttt{e},\texttt{f}),(\texttt{f},\texttt{d}),(\texttt{f},\texttt{e})\},\\
\texttt{at}^{I_R} &= \{\texttt{a}\},\text{ and}\\
\texttt{reachable}^{I_R} &= \{\texttt{a},\texttt{b},\texttt{c}\}.
\end{aligned}$$

Finally, the following program describes non-singleton cliques reachable from vertex a by `in_clique`, which is defined using `edge` and `reachable`:

$$\begin{aligned}
&\{\texttt{in_clique}(\texttt{X})\} \leftarrow \texttt{reachable}(X)\\
&\leftarrow \texttt{in_clique}(X),\ \texttt{in_clique}(Y),\ \mathit{not}\ \texttt{edge}(X,Y),\ X \neq Y\\
&\leftarrow \mathit{not}\ 2\{X : \texttt{in_clique}(X)\}.
\end{aligned} \tag{9}$$

In RASPL-1, expression $b\{\mathbf{x} : F(\mathbf{x})\}$, where b is a positive integer, $\mathbf{x}$ is a list of object variables, and $F(\mathbf{x})$ is a conjunction of literals, stands for the first-order formula

$$\exists \mathbf{x}^1 \dots \mathbf{x}^b \left[\bigwedge_{1\le i\le b} F(\mathbf{x}^i) \wedge \bigwedge_{1\le i<j\le b} \neg(\mathbf{x}^i = \mathbf{x}^j) \right],$$

where $\mathbf{x}^1,\dots,\mathbf{x}^b$ are lists of new object variables of the same length as $\mathbf{x}$. For any lists of variables $\mathbf{x} = (x_1,\dots,x_n)$ and $\mathbf{y} = (y_1,\dots,y_n)$ of the same length, $\mathbf{x} = \mathbf{y}$ stands for $x_1 = y_1 \wedge \dots \wedge x_n = y_n$.

The first-order module $\mathbf{F}_C$ is $(F_C, \{\texttt{reachable},\texttt{edge}\}, \{\texttt{in\_clique}\})$, where F_C is the following FOL-representation of RASPL-1 program (9):

$$\begin{aligned}
&\forall X(\texttt{reachable}(X) \rightarrow (\texttt{in_clique}(X) \vee \neg\texttt{in_clique}(X)))\\
&\quad \wedge \forall XY(\texttt{in_clique}(X) \wedge \texttt{in_clique}(Y) \wedge \neg\texttt{edge}(X,Y) \wedge X \neq Y\ \rightarrow\ \bot)\\
&\quad \wedge (\neg\exists XY(\texttt{in_clique}(X) \wedge \texttt{in_clique}(Y) \wedge X \neq Y) \rightarrow \bot)\ .
\end{aligned}$$

Let I_C be the following Herbrand $\mathbf{c}(F_C)$-partial interpretation of σ that satisfies

$\mathrm{SM}[\mathbf{F}_C]$, which is compatible with I_G and I_R.

$$\begin{aligned}\mathtt{edge}^{I_C} &= \{(\mathtt{a},\mathtt{a}),(\mathtt{a},\mathtt{b}),(\mathtt{b},\mathtt{c}),(\mathtt{c},\mathtt{b}),\\ &\qquad (\mathtt{c},\mathtt{c}),(\mathtt{d},\mathtt{e}),(\mathtt{d},\mathtt{f}),(\mathtt{e},\mathtt{d}),(\mathtt{e},\mathtt{f}),(\mathtt{f},\mathtt{d}),(\mathtt{f},\mathtt{e})\},\\ \mathtt{reachable}^{I_C} &= \{\mathtt{a},\mathtt{b},\mathtt{c}\},\ \text{and}\\ \mathtt{in\_clique}^{I_C} &= \{\mathtt{b},\mathtt{c}\}.\end{aligned}$$

Clearly, $\mathbf{F}_G$, $\mathbf{F}_R$, and $\mathbf{F}_C$ are joinable. In accordance with Theorem 4, the union of the partial interpretations $I_G \cup I_R \cup I_C$ is a partial interpretation of σ that satisfies $\mathrm{SM}[\mathbf{F}_G \sqcup \mathbf{F}_R \sqcup \mathbf{F}_C]$.

6 Modules That Can Be Incrementally Assembled

6.1 Review: Incremental Modularity by Gebser et al.

In this section, we present a reformulation of the theory behind the system ICLINGO, which was developed to allow for incremental grounding and solving of answer set programs. We follow the enhancement given in (Gebser et al. 2011) with a slight deviation. Most notably, we do not restrict attention to nondisjunctive logic programs, but limit attention to offline programs for simplicity.

Given a disjunctive program Π of a signature σ, by $\mathrm{Ground}_\sigma(\Pi)$ we denote the ground program obtained from Π by replacing object variables with ground terms in the Herbrand Universe of σ. If Π is ground, then the *projection* of Π onto a set X of ground atoms, denoted by $\Pi|_X$, is defined to be the program obtained from Π by removing all rules (3) in Π that contain some b_i not in X, and then removing all occurrences of *not* c_j such that c_j is not in X from the remaining rules. By $\mathit{head}(\Pi)$ we denote the set of all atoms that occur in the head of a rule in Π.

Definition 9 (*DLP-Module Instantiation*)
Given a disjunctive program Π, and a set of ground atoms $\mathcal{I}$, Gebser et al. (2011) define the *DLP-module instantiation* of Π w.r.t. $\mathcal{I}$, denoted by $DM(\Pi,\mathcal{I})$, to be the DLP-module $(\mathrm{Ground}_\sigma(\Pi)|_{\mathcal{I}\cup\mathcal{O}},\mathcal{I},\mathcal{O})$, where $\mathcal{O}$ is $\mathit{head}\big(\mathrm{Ground}_\sigma(\Pi)|_{\mathcal{I}\cup\mathit{head}(\mathrm{Ground}_\sigma(\Pi))}\big)$.

For example, Figure 2 shows a simple program and its DLP-module instantiation w.r.t. $\{l,t\}$.

An *incrementally parameterized program* $\Pi[t]$ is a program which may contain atoms of the form $a_{f(t)}(\mathbf{x})$, called *incrementally parameterized atoms*, where t is an *incremental step counter*, and $f(t)$ is some arithmetic function involving t. Given such a program $\Pi[t]$, its *incremental instantiation* at some nonnegative integer i, which we denote by $\Pi[i]$, is defined to be the program obtained by replacing all occurrences of atoms $a_{f(t)}(\mathbf{x})$ with an atom $a_v(\mathbf{x})$, where v is the result of evaluating

$$\left.\begin{array}{l} n \leftarrow t\\ p \leftarrow q, t\\ q \leftarrow r, \mathit{not}\ s\\ r \leftarrow m \end{array}\right\} \quad \underset{I=\{l,t\}}{\longmapsto} \quad \left(\begin{array}{l} n \leftarrow t\\ p \leftarrow q, t \end{array}, \{l,t\}, \{n,p,q\}\right)$$

Fig. 2. DLP-module instantiation of a simple program

$f(i)$. For example, let $\Pi = \{p_{t+1}(x) \leftarrow p_t(x), \mathit{not}\ q(x)\}$. The program $\Pi[2]$ is then $\{p_3(x) \leftarrow p_2(x), \mathit{not}\ q(x)\}$.

Gebser et al. (2011) define an *incremental logic program* to be a triple $\langle B, P[t], Q[t]\rangle$, where B is a disjunctive logic program, and $P[t]$, $Q[t]$ are incrementally parameterized disjunctive logic programs. Informally, B is the *base* program component, which describes static knowledge; $P[t]$ is the *cumulative* program component, which contains information regarding every step t that should be accumulated during execution; $Q[t]$ is the *volatile query* program component, containing constraints or information regarding the final step.

We assume a partial order $\prec$ on

$$\begin{aligned}\{&\mathrm{Ground}_\sigma(B),\mathrm{Ground}_\sigma(P[1]), \mathrm{Ground}_\sigma(P[2]), \dots,\\ &\mathrm{Ground}_\sigma(Q[1]), \mathrm{Ground}_\sigma(Q[2]), \dots\}\end{aligned} \tag{10}$$

such that

- $\mathrm{Ground}_\sigma(B) \prec \mathrm{Ground}_\sigma(P[1]) \prec \mathrm{Ground}_\sigma(P[2]) \prec \dots$;
- $\mathrm{Ground}_\sigma(P[i]) \prec \mathrm{Ground}_\sigma(Q[i])$ for $i \geq 1$.

Given a DLP-module $\mathbf{P} = (\Pi, \mathcal{I}, \mathcal{O})$, by $\mathit{Out}(\mathbf{P})$ we denote $\mathcal{O}$.

Definition 10 (*Modular and Acyclic Logic Programs*)
An incremental logic program $\langle B, P[t], Q[t]\rangle$ is *modular* if the following DLP-modules are defined for every $k \geq 0$:

$$\begin{aligned}\mathbf{P}_0 &= \mathit{DM}(B, \emptyset),\\ \mathbf{P}_i &= \mathbf{P}_{i-1} \sqcup \mathit{DM}(P[i], \mathit{Out}(\mathbf{P}_{i-1})), \qquad (1 \leq i \leq k)\\ \mathbf{R}_k &= \mathbf{P}_k \sqcup \mathit{DM}(Q[k], \mathit{Out}(\mathbf{P}_k)),\end{aligned}$$

and is *acyclic* if, for each pair of programs Π, Π' in (10) such that $\Pi \prec \Pi'$, we have that Π contains no head atoms of Π'. [3]

Gebser et al. (2011) demonstrated that given a modular and acyclic incremental logic program $\langle B, P[t], Q[t]\rangle$ and some nonnegative integer k, we are able to evaluate each component DLP-module individually, and compose the results in order to obtain the answer sets of the complete module $\mathbf{R}_k$. They define the *k-expansion* R_k of the incremental logic program as

$$B \cup P[1] \cup \dots \cup P[k] \cup Q[k]\ .$$

Proposition 2
(Gebser et al. 2011, Proposition 2) Let $\langle B, P[t], Q[t]\rangle$ be an incremental logic program of a signature σ that is modular and acyclic, let k be a nonnegative integer, and let X be a subset of the output atoms of $\mathbf{R}_k$. Set X is an answer set of the k-expansion R_k of $\langle B, P[t], Q[t]\rangle$ if and only if X is a (module) answer set of $\mathbf{R}_k$.

[3] The acyclicity condition corresponds to the special case of the "mutually revisable" condition in (Gebser et al. 2011) when there is no online component.

Proposition 2 tells us that the results of incrementally grounding and evaluating an incremental logic program are identical to the results of evaluating the entire k-expansion in the usual non-incremental fashion.

6.2 Incrementally Assembled First-Order Modules

In this section, we consider an extension of the theory supporting system ICLINGO which allows for the consideration of first-order sentences by utilizing Theorem 4. This extension may be useful in analyzing non-ground RASPL-1 programs that describe dynamic domains.

Given a first-order sentence F, we define the *projection* of F onto a set $\mathbf{p}$ of predicates, denoted by $F|_{\mathbf{p}}$, to be the first-order sentence obtained by replacing all occurrences of atoms of the form $q(t_1, \ldots, t_n)$ in F such that $q \in pr(F) \setminus \mathbf{p}$ with $\bot$ and performing the following syntactic transformations recursively until no further transformations are possible:

$$
\begin{array}{llll}
\neg\bot \mapsto \top & \neg\top \mapsto \bot & & \\
\bot \wedge F \mapsto \bot & F \wedge \bot \mapsto \bot & \top \wedge F \mapsto F & F \wedge \top \mapsto F \\
\bot \vee F \mapsto F & F \vee \bot \mapsto F & \top \vee F \mapsto \top & F \vee \top \mapsto \top \\
\bot \rightarrow F \mapsto \top & F \rightarrow \top \mapsto \top & \top \rightarrow F \mapsto F & \\
\exists x \top \mapsto \top & \exists x \bot \mapsto \bot & \forall x \top \mapsto \top & \forall x \bot \mapsto \bot
\end{array}
$$

For example, consider the first-order sentence

$$\forall x(p(x) \rightarrow q(x)) \wedge (q(a) \wedge \neg p(a) \rightarrow r) \wedge \forall x(\neg q(x) \wedge t(x) \rightarrow s(x)) \ . \qquad (11)$$

The projection of (11) onto $\{q, r, s, t, m\}$ is

$$(q(a) \rightarrow r) \wedge \forall x(\neg q(x) \wedge t(x) \rightarrow s(x)) \ .$$

When we restrict attention to the case of propositional logic programs such that $\mathbf{p}$ contains at least the predicates occurring strictly positively in F, this notion coincides with the corresponding one in the previous section.

Similar to incrementally parameterized programs, we define an *incrementally parameterized* formula $F[t]$ to be a first-order formula which may contain incrementally parameterized atoms. For any nonnegative integer i, we define the *incremental instantiation* of F at i, denoted by $F[i]$, to be the result of replacing all occurrences of incrementally parameterized atoms $a_{f(t)}(\mathbf{x})$ in $F[t]$ with an atom $a_v(\mathbf{x})$, where v is the result of evaluating $f(i)$.

Definition 11 (*First-Order Module Instantiation*)
For any first-order sentence F and any set of (input) predicates $\mathcal{I}$, formula F^0 is defined as F, and F^{i+1} is defined as $F^i|_{\mathcal{I} \cup head(F^i)}$, where $head(F^i)$ denotes the set of all predicates occurring strictly positively in F^i. We define the *first-order module instantiation* of F w.r.t. $\mathcal{I}$, denoted by $FM(F, \mathcal{I})$, to be the first-order module

$$(F^\omega, \ \mathcal{I}, \ pr(F) \backslash \mathcal{I}),$$

where F^ω is the least fixpoint of the sequence $F^0, F^1, \ldots$.

The idea of the simplification process is related to the fact that all predicates other than the ones in $\mathcal{I} \cup head(F^i)$ have empty extents under the stable model semantics, which are equivalent to $\bot$ (Ferraris et al. 2011, Theorem 4). The process is guaranteed to lead to a fixpoint in a finite number of steps since F is finite and $F^i|_{\mathcal{I} \cup head(F^i)}$ is shorter than F^i in all cases except for the terminating case. It is not difficult to check that if F is the FOL-representation of a ground disjunctive program Π, the first component $\text{Ground}_\sigma(\Pi)|_{\mathcal{I} \cup \mathcal{O}}$ in the definition of a DLP-module instantiation corresponds to F^2.

Example 2
Consider the propositional formula

$$F = (p \to q) \wedge (q \to r) \wedge (t \wedge \neg r \to s)\ .$$

and $\mathcal{I} = \{t, m\}$. The process of instantiation results in the following transformations on F:

$$\begin{array}{lr}
(p \to q) \wedge (q \to r) \wedge (t \wedge \neg r \to s). & F^0 (= F) \\
\Rightarrow\ (q \to r) \wedge (t \wedge \neg r \to s). & F^1 \\
\Rightarrow\ t \wedge \neg r \to s. & F^2 \\
\Rightarrow\ t \to s. & F^3 \\
\Rightarrow\ t \to s. & F^4
\end{array}$$

The resulting first-order module is then

$$FM(F, \{t, m\}) = (t \to s,\ \ \{t, m\},\ \ \{p, q, r, s\}).$$

This definition of an instantiation is different from the one by Gebser et al. (2011) even when we restrict attention to a finite propositional disjunctive program. First, we maximize the simplification done on the initial formula F by repeatedly projecting it onto its head and input predicates, whereas Gebser et al. perform only the first two projections (i.e., F^2). Second, the list of output atoms are different. In our case all atoms occurring in F that are not input atoms are assumed to be output atoms. The following example illustrates these differences.

Example 3
Recall the DLP-module instantiation in Figure 2. The first-order instantiation of (the FOL-representation of) the program w.r.t $\{l, t\}$ is $(t \to n,\ \{l, t\},\ \{m, n, p, q, r, s\})$.

While the two notions of instantiation are syntactically different, it can be shown that, given a propositional logic program Π and sets of propositional atoms I and X, X is a module answer set of $DM(\Pi, I)$ if and only if X is a module answer set of $FM(\Pi, I)$.

An *incremental first-order theory* is a triple $\langle B, P[t], Q[t] \rangle$ where B is a first-order sentence, and $P[t]$ and $Q[t]$ are incrementally parameterized sentences.

The *k-expansion* of $\langle B, P[t], Q[t] \rangle$ is defined as

$$R_k = B \wedge P[1] \wedge \cdots \wedge P[k] \wedge Q[k].$$

It is clear that this coincides with the notion of k-expansion for incremental logic programs when we restrict attention to the common syntax.

We assume a partial order $\prec$ on

$$\{B, P[1], P[2], \dots, Q[1], Q[2], \dots\} \tag{12}$$

as follows:

- $B \prec P[1] \prec P[2] \prec \dots$;
- $P[i] \prec Q[i]$ for $i \geq 1$.

Definition 12 (*Acyclic Incremental First-Order Theory*)
We say that an incremental first-order theory $\langle B, P[t], Q[t]\rangle$ is *acyclic* if, for every pair of formulas F, G in (12) such that $F \prec G$, we have that G is negative on $pr(F)$.

This definition of acyclicity mirrors that of Gebser *et al.*'s (2011) in that it prevents predicates from occurring strictly positively in multiple sentences which are instantiated from the incremental theory. However, as shown in Proposition 3, it is unnecessary to check a condition similar to modularity for incremental first-order theories, as it is ensured by acyclicity.

Given a first-order module $\mathbf{F} = (F, \mathcal{I}, \mathcal{O})$, by $Out(\mathbf{F})$ we denote $\mathcal{O}$.

Proposition 3 (*Modularity of Incremental Theory*)
If an incremental first-order theory $\langle B, P[t], Q[t]\rangle$ is acyclic, then the following modules are defined for all $k \geq 0$.

$$\begin{aligned}
\mathbf{P}_0 &= FM(B, \emptyset), \\
\mathbf{P}_i &= \mathbf{P}_{i-1} \sqcup FM(P[i], Out(\mathbf{P}_{i-1})), && (1 \leq i \leq k) \\
\mathbf{R}_k &= \mathbf{P}_k \sqcup FM(Q[k], Out(\mathbf{P}_k)) \ .
\end{aligned}$$

By applying Theorem 4, we can evaluate each component module independently and compose their results in order to obtain the stable models of $\mathbf{R}_k$.

Proposition 4 (*Compositionality for Incremental First-Order Theories*)
Let $\langle B, P[t], Q[t]\rangle$ be an incremental first-order theory and let $\mathbf{R}_k$ be the module as defined in the statement of Proposition 3. For any nonnegative integer k,

$$\begin{aligned}
I_B \cup I_{P[1]} \cup \dots \cup I_{P[k]} \cup I_{Q[k]} \models \mathrm{SM}[\mathbf{R}_k] \\
\text{iff} \quad I_B \models \mathrm{SM}[FM(B, \emptyset)] \\
\text{and} \quad I_{P[1]} \models \mathrm{SM}[FM(P[1], Out(\mathbf{P}_0))] \\
\text{and} \quad \dots \\
\text{and} \quad I_{P[k]} \models \mathrm{SM}[FM(P[k], Out(\mathbf{P}_{k-1}))] \\
\text{and} \quad I_{Q[k]} \models \mathrm{SM}[FM(Q[k], Out(\mathbf{P}_k))] \ .
\end{aligned} \tag{13}$$

where I_B ($I_{P[1]}, \dots, I_{P[k]}, I_{Q[k]}$, respectively) is a $\mathbf{c}(B)$-partial interpretation ($\mathbf{c}(P[1])$, $\dots, \mathbf{c}(P[k]), \mathbf{c}(Q[k])$-partial interpretation, respectively) such that $I_B, I_{P[1]}, \dots, I_{P[k]}, I_{Q[k]}$ are pairwise compatible.

Given an acyclic incremental theory and a nonnegative integer k, the following proposition states that evaluating the individual modules and composing their results is equivalent to evaluating the k-expansion of the incremental theory.

Proposition 5 (*Correctness of Incremental Assembly*)
Let $\langle B, P[t], Q[t] \rangle$ be an acyclic incremental theory, let k be a nonnegative integer, let R_k be the k-expansion of the incremental theory, and let $\mathbf{R}_k$ be the module as defined in Proposition 3. For any $\mathbf{c}$-partial interpretation I such that $\mathbf{c} \supseteq \mathbf{c}(R_k)$, we have that

$$I \models \mathrm{SM}[R_k] \text{ iff } I \models \mathrm{SM}[\mathbf{R}_k].$$

7 Conclusion

Our extension of the module theorem to the general theory of stable models applies to non-ground logic programs containing choice rules, the count aggregate, and nested expressions. The extension is based on the new findings about the relationship between the module theorem and the splitting theorem. The proof of our module theorem[4] uses the splitting theorem as a building block so that a further generalization of the splitting theorem can be applied to generalize the module theorem as well. Indeed, the module theorem presented here can be extended to logic programs with arbitrary (recursive) aggregates, based on the extension of the splitting theorem to formulas with generalized quantifiers, recently presented in (Lee and Meng 2012). Based on the generalized module theorem, we reformulated and extended the theory of incremental answer set computation to the general theory of stable models, which can be useful in analyzing non-ground RASPL-1 programs that describe dynamic domains.

Acknowledgements

We are grateful to Martin Gebser and Tomi Janhunen for useful discussions related to this paper. We are also grateful to the anonymous referees for their useful comments. This work was partially supported by the National Science Foundation under Grant IIS-0916116.

References

Ferraris, P., Lee, J., and Lifschitz, V. 2007. A new perspective on stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 372–379.

Ferraris, P., Lee, J., and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence 175*, 236–263.

Ferraris, P., Lee, J., Lifschitz, V., and Palla, R. 2009. Symmetric splitting in the general theory of stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 797–803.

Gebser, M., Grote, T., Kaminski, R., and Schaub, T. 2011. Reactive answer set programming. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Springer, 54–66.

[4] Available in the online appendix.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Thiele, S. 2008. Engineering an incremental ASP solver. In *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*, M. Garcia de la Banda and E. Pontelli, Eds. Lecture Notes in Computer Science, vol. 5366. Springer-Verlag, 190–205.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of International Logic Programming Conference and Symposium*, R. Kowalski and K. Bowen, Eds. MIT Press, 1070–1080.

Janhunen, T., Oikarinen, E., Tompits, H., and Woltran, S. 2009. Modularity aspects of disjunctive stable models. *Journal of Artificial Intelligence Research 35*, 813–857.

Lee, J., Lifschitz, V., and Palla, R. 2008. A reductive semantics for counting and choice in answer set programming. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 472–479.

Lee, J. and Meng, Y. 2012. Stable models of formulas with generalized quantifiers. In *Proceedings of International Workshop on Nonmonotonic Reasoning (NMR)*. `http://peace.eas.asu.edu/joolee/papers/smgq-nmr.pdf`.

Lee, J. and Palla, R. 2012. Reformulating the situation calculus and the event calculus in the general theory of stable models and in answer set programming. *Journal of Artificial Inteligence Research (JAIR) 43*, 571–620.

Lifschitz, V. 1994. Circumscription. In *Handbook of Logic in AI and Logic Programming*, D. Gabbay, C. Hogger, and J. Robinson, Eds. Vol. 3. Oxford University Press, 298–352.

Oikarinen, E. and Janhunen, T. 2008. Achieving compositionality of the stable model semantics for smodels programs. *TPLP 8,* 5-6, 717–761.