Extensions of logic programming languages with structuring constructs have been extensively studied in the last years. A particularly clean approach consists in allowing implication goals to occur in goals and in clause bodies. Implication goals are implications of the form $D \Rightarrow G$, where $D$ is a set of clauses and $G$ is a goal. The clauses in $D$ are intended to be local to $G$, as they can be used only in the proof of $G$. So implication goals can be regarded as blocks of conventional programming languages and such an extension provides different kinds of block structured languages according to the visibility rules chosen for the local clauses. The choices are mainly two: either lexical (static) visibility rules or dynamic visibility rules. A further distinction can be made between closed and open blocks. In this paper we aim at showing that modal logic provides a unifying framework in which different logical languages with blocks and modules can be expressed. We show this by interpreting the different block languages within modal logic S4. Moreover, we will consider different kinds of modules that can be defined from blocks by introducing some syntactic sugar and compare them to other proposals in the literature.

1. INTRODUCTION

The need for structuring constructs to extend logic programs has produced a large amount of proposals. Though some of them can be regarded as mainly syntactic extensions, others have a clean and logical semantics. Among the latter, a particularly clean approach for introducing local definitions of clauses consists in allowing implication goals to occur nested in goals and in clause bodies. Implication goals are implications of the form $D \Rightarrow G$, where $D$ is a set of clauses and $G$ is a goal.
The clauses in $D$ are intended to be local to $G$, as they can be used only in the proof of $G$. In this way, a goal $G_i$ in a clause $G_1 \land \cdots \land G_n \rightarrow A$ can be not only an atom, but also an implication goal. Since implication goals introduce local clause definitions, we will call them blocks, by analogy with conventional Algol-like programming languages.

In [14, 15] such an extension is introduced to deal with hypothetical reasoning and the operational semantics of the extended language is proved to be sound and complete w.r.t intuitionistic logic (by regarding both $\rightarrow$ and $\Rightarrow$ as the intuitionistic implication). For this reason, it is referred to as intuitionistic logic programming. In [25, 26] a superset of this language is considered, containing also the disjunction in goals, and a fixpoint semantics is defined for it. Intuitionistic logic programming has also been studied in [23, 24]. An analysis of these proposals shows that they use dynamic visibility rules for clauses, since, operationally, to prove a goal $D - G$ in a program $P$ they prove the goal $G$ in the extended program $P \cup D$ (and thus $P$ can use clauses in $D$ and vice versa).

However, the same extended language with implication goals can be given a different operational semantics if different visibility rules for locally defined clauses are chosen. In particular, in [17, 18, 30] languages with static scope rules for clause definitions have been defined, in which, as in Algol-like languages, the rules for using clauses are determined by the lexical structure of the program ($D$ can use clauses in $P$, but not vice versa). Furthermore, in [18] a notion of closed block is defined (as opposed to open block) which mimics at the object level the metapredicate demo of Bowen and Kowalski [5], so that $D \Rightarrow G$, roughly speaking, can be read as $\text{demo}(D, G)$.

In this paper we want to show that these different languages with blocks can all be given a modal characterization in the logic $S_4$. The modal interpretation of the language with dynamic visibility rules is the obvious one, since such a language, as proved in [15, 26], has an intuitionistic semantics, and intuitionistic logic can be interpreted within $S_4$ modal logic. A modal characterization for the language with static blocks and closed blocks can be obtained by slight modification.

As a consequence, a modal extension of Horn clause logic can provide a unifying framework in which different kinds of blocks can be defined and integrated. The possibility of integrating different kinds of blocks is given by the similarity of their modal characterizations. In fact, the difference between the languages with static and dynamic visibility rules does not reside in how the implication goal is defined, but in how clauses (both the local and the global) are defined. Thus, by allowing different kinds of clause definitions (either static or dynamic), an integrated language can be defined.

We want to make clear that our main concern in this paper is not to study how logic programming can be extended to modal logic programming in the general case. Rather, we will focus on the problem of structuring logic programs and show that a modal extension of logic programming is well suited to this purpose.

The outline of the paper is the following. In Section 2 we recall the definition of the different languages with blocks and their operational semantics. Section 3 presents a modal characterization within $S_4$ of these languages in the propositional case. In Section 4 such a characterization is extended to the first-order case and its correctness is proved for the language with open static blocks. Moreover, it is shown that, without loss of generality, we can restrict our concern to Kripke interpretations in which the Herbrand universe is the constant domain of quantifi-
cation for each world. In Section 5 the operational semantics of an integrated language is defined and some examples of use of the language are provided. In particular, we show how modules also can be defined, by introducing some syntactic sugar, and we compare them to other proposals for blocks and modules in the literature. A final section is devoted to conclusions and related work.

2. BLOCK LANGUAGES WITH DIFFERENT VISIBILITY RULES

In this section we recall the syntax and the operational semantics of the different block languages. As proposed in [18], we make a distinction among open blocks with dynamic visibility rules [14, 25], open blocks with static visibility rules [17, 18, 30], and closed blocks [18].

All of these block languages are obtained by allowing implication goals of the form $D \Rightarrow G$ both in goals and in clause bodies. When a goal $D \Rightarrow G$ has to be proved, the goal $G$ must be proved in the current context (program) extended with the local set of clauses $D$. According to the way the current context is extended with $D$, different kinds of block structured logic languages can be obtained. Though their syntax is the same, they differ as regards their operational semantics.

Let $x$ represent an individual variable. Let $A$ be a first-order atomic formula and let $T$ be a propositional constant (true). The syntax of the languages with blocks is the following:

$$L: \quad G := T | A | G_1 \land G_2 | \exists x G | D \Rightarrow G,$$

$$D := G \Rightarrow A | D_1 \land D_2 | \forall x D.$$

In this definition $G$ stands for a goal and $D$ stands for a clause or a conjunction of clauses. In the following text, $D$ will be regarded interchangeably as a conjunction or as a set of clauses. A program is a set of clauses.

Note that in a goal $D \Rightarrow G$ it is possible for $D$ and $G$ to share variables, like, for instance, in the goal $\exists x [(T \rightarrow p(x)) \Rightarrow q(x)]$.

Contrary to [26] we have introduced two different implications in goals and in clauses. In fact, though in the dynamic case both implications have the same semantics (they are both the intuitionistic implication), in the static case the two implications are different.

2.1. Open Blocks with Dynamic Scoping

According to the analysis in [18], the languages in [14, 25, 26] can be regarded as languages with open blocks and dynamic visibility rules. In this section we recall their operational semantics following [26] (as a difference, we do not consider disjunctions in goals, whose treatment, however, is easy).

In order to avoid problems with variable renaming and substitutions, given a program $P$, we define a set of clauses $[P]$, which contains $P$ along with all instances of clauses in $P$ obtained by replacing universally quantified variables in front of them with ground terms. More precisely, $[P]$ can be defined recursively as the smallest set of formulas such that:

1. $P \subseteq [P]$;
2. if $D_1 \land D_2 \in [P]$, then $D_1 \in [P]$ and $D_2 \in [P]$;
3. if $\forall x D \in [P]$, then $[x/t]D \in [P]$ for all closed terms $t$. 
Given a program $P$ and a closed goal $G$, the meaning of $G$ being operationally derivable from $P$, that is $P \vdash G$, is defined by induction on the structure of $G$ as follows:

- $P \vdash T$;
- $P \vdash A$ if there is a formula $G \rightarrow A \in [P]$ and $P \vdash G$;
- $P \vdash G_1 \land G_2$ if $P \vdash G_1$ and $P \vdash G_2$;
- $P \vdash \exists x G$ if there is a closed term $t$ such that $P \vdash [t/x]G$;
- $P \vdash D \Rightarrow G$ if $P \cup D \vdash G$.

A derivation of $G$ from $P$ is a sequence of pairs $(W_1, G_1), \ldots, (W_m, G_m)$, where $W_1 = P$, $G_1 = G$, $G_m = T$, and, for $i = 1, \ldots, m$, $W_i \vdash G_i$. Derivability of $G_i$ from $P_i$ can be concluded from the pairs which follow $(W_i, G_i)$ in the sequence, using the rules above.

The visibility rules for clauses in this block language are dynamic because the set of clauses that can be used to solve a goal $G$ depends on the sequence of goals generated up to that point and can be determined only dynamically. In fact, an implication goal can simply be regarded as specifying a query in an updated program.

**Example 2.1.** The proof of the goal $G = s$ in the program

$P: \quad r \rightarrow q$

$$(((q \rightarrow p) \land r) \Rightarrow p) \rightarrow s$$

succeeds, since it amounts to proving

- goal $((q \rightarrow p) \land r) \Rightarrow p$ in $P$,
- goal $p$ in $P' = P \cup \{q \rightarrow p, r\}$,
- goal $q$ in $P'$,
- goal $r$ in $P'$,

which succeeds. The proof of $q$ uses the clause $r$ defined in the inner block, which is visible at that point since the block has been added to the program $P$. If, on the contrary, the goal $q$ is called directly from the outer environment, its proof fails.

With this operational behavior, open blocks with dynamic scoping are well suited for hypothetical reasoning. Moreover, as shown in [26], by giving names to sets of clauses (for instance, by writing $m \equiv D$) and by using those names within implication goals (as in $m \Rightarrow G$), modules can be defined. Also, nesting implication goals allows modules to be composed. For instance, by means of the query $m_1 \Rightarrow (m_2 \Rightarrow G)$, $G$ is proved in the union of the two modules $m_1$ and $m_2$.

### 2.2. Open Blocks with Static Scoping

Given the above language $L$ with implication goals, static scope rules for local definitions of clauses can be obtained by modifying the previously given operational semantics (as done in [17, 18]). In this case, we want the rules for using a clause to be determined only by the static nesting of blocks in the program text. To solve a
goal coming from the body of a clause declared in a block, we want to use only the clauses declared in that block or in externally nested blocks. With this choice, we fail in proving the goal \( a \Rightarrow b \) from the program \( P = \{ a \Rightarrow c, c \Rightarrow b \} \), since the atom \( a \) introduced to prove \( b \) is local to the goal and cannot be used to solve the subgoal \( a \) coming from the body of clause \( a \Rightarrow c \). For this reason, when static scope rules are chosen, implication goals cannot be used to model the updating of a set of clauses nor to perform hypothetical reasoning. Static scope rules provide structuring constructs which are closer to those of conventional (Algol-like) block structured programming languages.

Let us define the operational semantics for the language with static scoping. In this case it is necessary to consider the derivability of a goal \( G \) from a list of programs \( D_1 \cdots |D_n \), i.e., a list of sets of clauses. The list allows us to record the ordering between blocks given by their lexical nesting in the program text. The higher is the index \( i \) of \( D_i \) in \( D_1 \cdots |D_n \), the deeper is the nesting of \( D_i \) in the program. We can now define the derivability of a goal \( G \) from a list of programs \( D_1 \cdots |D_n \) by induction on the structure of \( G \) as follows:

\[
\begin{align*}
D_1 \cdots |D_n & \vdash T; \\
D_1 \cdots |D_n & \vdash A \text{ if, for some } i, 1 \leq i \leq n, \text{ there is a formula } G \Rightarrow A \in [D_i] \text{ and } D_1 \cdots |D_i \vdash G; \\
D_1 \cdots |D_n & \vdash G_1 \land G_2 \text{ if } D_1 \cdots |D_n \vdash G_1 \text{ and } D_1 \cdots |D_n \vdash G_2; \\
D_1 \cdots |D_n & \vdash \exists x G \text{ if there is a closed term } t \text{ such that } D_1 \cdots |D_n \vdash [t/x] G; \\
D_1 \cdots |D_n & \vdash D \Rightarrow G \text{ if } D_1 \cdots |D_n \vdash D \Rightarrow G.
\end{align*}
\]

Notice that when a clause \( G \Rightarrow A \) in \( D_i \) is used to prove an atomic goal \( A \), then the clauses in \( D_{i+1} \cdots |D_n \) cannot be used any more to prove \( G \), since the clauses of inner blocks are not visible from external ones (and thus from \( G \)). To prove a goal \( D \Rightarrow G \), the set of local clauses \( D \) is added to the list of programs as the tail element, so that the clauses in \( D \) can be used only to prove goals coming from \( D \) itself or from \( G \).

It is easy to see that, with this operational semantics, given the program \( P \) and the goal \( G \) of Example 2.1, \( G \) is not provable in \( P \). The following is an example of successful derivation.

**Example 2.2.** The proof of the goal \( G = s \) in the program

\[
P: \quad q \\
( ((r \land q \Rightarrow p) \land r) \Rightarrow p ) \Rightarrow s
\]

succeeds, since the local clauses \( (r \land q \Rightarrow p) \) and \( r \), introduced by proving the implication goal \( ((r \land q \Rightarrow p) \land r) \Rightarrow p \), are used only locally to the block.

With open static blocks, it is possible to compose different sets of clauses (different modules) while retaining a distinction between them. Consider, for instance, the query \( m_1 \Rightarrow (m_2 \Rightarrow (m_3 \Rightarrow G)) \), where the \( m_i \)'s are names for sets of clauses as in the previous paragraph. The goal \( G \) is proved in the composition of the modules \( m_1, m_2, \) and \( m_3 \), where \( m_2 \) can see all definitions in \( m_3 \), and \( m_3 \) can see all definitions in \( m_1 \) and \( m_2 \), but not vice versa. In this case, in composing modules, inheritance applies only in one direction.
2.3. Closed Blocks

There is at least one other way to define the operational semantics of the language \( L \) with implication goals. It is possible to regard an implication goal \( D \Rightarrow G \) as specifying a change of context: to prove the goal \( D \Rightarrow G \) from the program \( P \), prove \( G \) from the program \( D \) (the set of local clauses). For this reason the goal \( D \Rightarrow G \) can be regarded as mimicking the metapredicate Demo\((D,G)\) defined by Bowen and Kowalski [5]. Notice, however, that while in Demo\((D,G)\) \( D \) and \( G \) can be metavariables, this is not the case in an implication goal \( D \Rightarrow G \).

The derivability of a closed goal \( G \) from a program \( P \) is defined by induction on the structure of \( G \) as follows (we do not need a list of programs in this case):

- \( P \vdash T \);
- \( P \vdash A \) if there is a formula \( G \rightarrow A \in [P] \) and \( P \vdash G \);
- \( P \vdash G_1 \land G_2 \) if \( P \vdash G_1 \) and \( P \vdash G_2 \);
- \( P \vdash \exists x G \) if there is a closed term \( t \) such that \( P \vdash [t/x]G \);
- \( P \vdash D \Rightarrow G \) if \( D \vdash G \).

The idea of regarding an implication goal as specifying a change of context seems to be quite natural. When names are associated with sets of clauses, it provides a notion of module which is quite close to the conventional notion of module as a closed environment.

3. MODAL INTERPRETATION OF THE BLOCK LANGUAGES: THE PROPOSITIONAL CASE

In this section we give the S4 modal interpretation of the languages with blocks defined in the previous section. For simplicity, we will start by considering the propositional case, in which the differences between the block languages are already present in a simpler setting. We will try to give here some intuitive motivations to explain why the modal interpretation works. The first-order case will be dealt with in the next section.

3.1. Open Blocks with Dynamic Scoping

As regards the language with dynamic blocks, in [15, 26] it is proved that the operational semantics presented in Section 2.1 is sound and complete w.r.t. intuitionistic logic when the two implications \( \Rightarrow \) and \( \rightarrow \) are regarded as the same implication: the intuitionistic one. Therefore, such a language has the semantics of intuitionistic logic and the well known mapping from the language of intuitionistic propositional logic to the language of propositional S4 modal logic (see [13]) can be applied to it. We recall this mapping (denoting it by \( * \)) on the propositional formulas of the language \( L \):

\[
T^* = T,
A^* = \Box A \quad (A \text{ is an atomic proposition}),
(\alpha \land \beta)^* = \alpha^* \land \beta^*,
(\alpha \Rightarrow \beta)^* = \Box(\alpha^* \supset \beta^*),
(\alpha \rightarrow \beta)^* = \Box(\alpha^* \triangleright \beta^*),
\]
where \( \supset \) is the material implication and \( \Box \) is the universal modal operator. Of course, the two different implications are translated in the same way, since both of them stand for the intuitionistic implication.

By applying this mapping, the propositional part of language \( L \) can be translated into the following language \( L_1 \) in S4 modal logic:

\[
L_1: \quad G := T \Box A \land G_1 \land G_2 \Box (D \supset G), \\
D := \Box (G \supset \Box A) \land D_1 \land D_2.
\]

To give an intuitive reading of this modal language, let us recall the Kripke semantics for S4 modal logic (see [20]). By now we are only interested to the propositional case. However, we define first-order Kripke interpretations with the Herbrand universe as fixed domain, since we will need such definition in the following text, when dealing with the first-order case.

Let \( L_M \) be a first-order modal language containing the logical connectives \( \land, \supset, \exists, \) and \( \forall \) and the modal operator \( \Box \). Let \( U_L \) be the Herbrand universe of the language \( L_M \), that is, the set of ground terms formed out of the constants and function symbols of \( L_M \), and let \( H_L \) be the Herbrand base of \( L_M \), that is, the set of all ground atoms that can be formed using predicate symbols in \( L_M \) and terms in \( U_L \). An (Herbrand) S4-Kripke interpretation for \( L_M \) is a triple \( M = \langle W, R, e \rangle \), where \( W \) is a set of worlds, \( R \) is a binary relation on \( W \) (the accessibility relation) which is reflexive and transitive, and \( e \) is a valuation function \( e: W \rightarrow \mathcal{P}(H_L) \). We define the satisfiability of a closed formula \( \alpha \) of \( L_M \) at a world \( w \in W \) in an S4-Kripke interpretation \( M \) for \( L_M \) (\( M, w \vDash_{S4} \alpha \)) as follows:

- \( M, w \vDash_{S4} T \)
- \( M, w \vDash_{S4} A \) iff \( A \in e(w) \) (if \( A \) is an atom in \( H_L \))
- \( M, w \vDash_{S4} \alpha \land \beta \) iff \( M, w \vDash_{S4} \alpha \) and \( M, w \vDash_{S4} \beta \)
- \( M, w \vDash_{S4} \alpha \supset \beta \) iff \( M, w \not\vDash_{S4} \alpha \) or \( M, w \vDash_{S4} \beta \)
- \( M, w \vDash_{S4} \exists x \alpha \) iff for some closed term \( t \in U_L \), \( M, w \vDash_{S4} \alpha[t/x] \)
- \( M, w \vDash_{S4} \forall x \alpha \) iff for all closed terms \( t \in U_L \), \( M, w \vDash_{S4} \alpha[t/x] \)
- \( M, w \vDash_{S4} \Box \alpha \) iff for all worlds \( w' \in W \) such that \( Rw', M, w' \vDash_{S4} \alpha \).

A closed formula \( \alpha \) of \( L_M \) is true in an S4-Kripke interpretation \( M = \langle W, R, e \rangle \) iff \( M, w \vDash_{S4} \alpha \), for all \( w \in W \). We say that \( \alpha \) is a S4-valid formula (\( \vDash_{S4} \alpha \)) iff it is true in every S4-Kripke interpretation for \( L_M \), i.e., for every S4-Kripke interpretation \( M = \langle W, R, e \rangle \) (for \( L_M \)), for every \( w \in W \), \( M, w \vDash_{S4} \alpha \).

The correspondence between the language \( L \) with dynamic blocks and its modal interpretation \( L_1 \) can be stated as follows: for all programs \( P \) and goals \( G \),

\[
P \vdash G \iff \vDash_{S4} P^* \supset G^*.
\]

according to the definition of operational derivability given in Section 2.1.

By the semantics of S4 modal logic, an implication goal \( \Box (D \supset G) \) is true in an interpretation \( M \) at a world \( w \) if, in every world \( w' \) reachable from \( w \) in which \( D \) is true, \( G \) is also true. We can see the modal operator \( \Box \) as specifying a change of world from the current world to a new reachable world. This makes clear how an implication goal \( \Box (D \supset G) \) can be regarded as specifying the proof of the goal \( G \) in a new reachable world, a generic world of which it is only known that it satisfies \( D \).
Besides, when moving to a new world, the clauses present in the initial program still hold. In fact, in the language $L_1$, a program contains clauses of the form

$$\Box (G \supset \Box A)$$

and (by the transitivity of the accessibility relation in the logic S4) if a clause $\Box (G \supset \Box A)$ is true in an interpretation $M$ at a world $w$, then it is true in that interpretation at any world $w'$ reachable from $w$.

Thus, if we go from the world $w$ to the world $w'$ by updating the database, all the clauses of the program will still be available in $w'$. This models the fact that, operationally, in the case of dynamic scoping rules, the clauses in the global program are always available for further inferences when new clauses are introduced by an update. In particular, the clauses added by an implication goal (and coming from a certain block) can always be used to prove subgoals coming from the body of a clause defined in an external block. In other words, when a clause $\Box (G \supset \Box A)$ is entered in the database (by an update), then the possibility of proving its body $G$ is affected by the successive updates.

As an example, consider the program $P = \{a \rightarrow b\}$ and the goal $G = a \cdot b$. $G$ is operationally provable in the program $P$. In fact, when $a$ is added to the program then $b$ is derivable. In the modal interpretation, the program and the goal become, respectively,

$$P^* = \{ \Box (\Box a \supset \Box b)\} \quad \text{and} \quad G^* = \Box (\Box a \supset \Box b).$$

Therefore, it is quite obvious that $\models_{S4} P^* \supset G^*$.

### 3.2. Open Blocks with Static Scoping

As for open dynamically scoped blocks, we want to define an S4 modal interpretation for the propositional subset of the language with static scope rules. To this purpose, we define the following mapping $*$ on the propositional subset of $L$:

$$T^* = T,$$

$$A^* = \Box A \quad (A \text{ is an atomic proposition}),$$

$$(a \land \beta)^* = a^* \land \beta^*,$$

$$(a \rightarrow \beta)^* = \Box (a^* \supset \beta^*),$$

$$(a \rightarrow \beta)^* = a^* \supset \beta^*.$$

The difference w.r.t. to the previous mapping, which gives the language $L_1$, is that the two implications (in goals and in clauses) are interpreted in two different ways. The modal operator $\Box$ is not put in front of the implications in clause definitions. By applying this mapping to the propositional part of $L$, we get the following language:

$$L_2: \quad G := T\Box A | G_1 \land G_2 | \Box (D \supset G),$$

$$D := G \supset \Box A | D_1 \land D_2.$$
in front of clauses, while implication goals are interpreted in both cases in the same way.

This makes clear that, at least in this modal reconstruction, the difference between the two languages does not reside in how the implication goal is defined, but in how clauses (both the local and the global) are defined. Of course, since the definitions of clauses and goals are mutually recursive, the implication goals in the languages \( L_1 \) and \( L_2 \) also are different: they differ w.r.t. their local clause definitions.

By considering the possible world semantics for S4 modal logic, the difference between the languages \( L_1 \) and \( L_2 \) becomes quite intuitive. We said that the modal operator specifies a change of world and an implication goal \( \Box(D \supset G) \) is true in an interpretation \( M \) at a world \( w \) if \( G \) is true at all reachable worlds \( w' \) at which \( D \) holds. Since in the language \( L_2 \) the clauses of a program have the form \( G \supset \Box A \), then it does not hold that if a clause is true in an interpretation \( M \) at the world \( w \), then it is also true at any world \( w' \) reachable from \( w \), for clauses have no modal operator in front of them.

Thus, if we go from a world \( w \) to a reachable world \( w' \) by updating the database, the clause \( G \supset \Box A \), true in \( w \), may not be true in \( w' \). In fact, in the case of static scoping, the clauses in the global program are not available for further inferences when we move to a reachable world, i.e., when new clauses are introduced by an update. The clauses added by an implication goal (and coming from a certain block) cannot be used to prove subgoals coming from the body of a clause defined in an external block.

Only the atomic consequences of clauses in the global program remain available in all the reachable worlds and, therefore, clauses in the program can be used to prove the body of a clause in a block, but not vice versa. In particular, if the clause \( G \supset \Box A \) occurs in the initial program, then proving \( G \) cannot be affected by successive updates to the program by implication goals: the clauses introduced by the successive updates cannot be used to prove \( G \).

Therefore, the language \( L_2 \) is a very static language in which, given a program, the updates occurring in the goal or in the program itself cannot have much influence on what is derivable from the initial program. A block allows new atoms to be derived by introducing new clauses (procedures), but it does not affect the initial program.

In Section 4 we will prove that the above interpretation of the propositional language with open static blocks within S4 modal logic is correct, i.e., we will prove that, for all programs \( P \) and goals \( G \),

\[
P \vdash G \quad \text{iff} \quad \vdash_{s4} P^* \supset G^*,
\]

where \( \vdash \) is operational derivability relation as defined in Section 2.2. The correctness proof will be provided for the first-order case in Section 4.

Let us give here just an example. Consider again the program \( P = \{ a \rightarrow b \} \) and the goal \( G = a \Rightarrow b \). If static visibility rules are adopted, \( G \) is not operationally provable in the program \( P \). In fact, to prove \( a \Rightarrow b \), \( a \) is added to the program, but it cannot be used to prove the goal \( a \) coming from the body of the clause \( a \rightarrow b \) in the global program since it is not visible. In the modal interpretation, the program and the goal are

\[
P^* = \{ \Box a \supset \Box b \} \quad \text{and} \quad G^* = \Box( \Box a \supset \Box b),
\]
and it is not true that $\models_{s4} P^* \supset G^*$. In fact, if we take the S4 interpretation $M = \langle (w_1, w_2), R, e \rangle$, where $R = \{(w_1, w_2), (w_1, w_1), (w_2, w_2)\}$ and $e(w_1) = \{1\}$ and $e(w_2) = \{a\}$, then $M, w_1 \models_{s4} \Box a \supset \Box b$, but $M, w_1 \not\models_{s4} \Box (\Box a \supset \Box b)$.

3.3. Closed Blocks

The language of closed blocks also can be interpreted within S4-modal logic. In [18] a Kripke-like model-theoretic semantics is defined for this language and, making use of it, it is possible to prove that the language of closed blocks can be interpreted within S4-modal logic as follows:

$L_3$: $G := T | A | G_1 \land G_2 \land (D \supset G)$,
$D := G \supset A | D_1 \land D_2$.

The difference w.r.t. the language $L_2$ is that in this case there is no modal operator in front of atomic formulas. As for the language $L_2$, there is no modal operator in front of clause definitions. Therefore, the only occurrence of the modal operator is the one in front of implications in goals.

For this reason, intuitively, the modal operator has the effect of closing a context: there is neither a clause nor an atom which has a box in front of it and can pass through the modal context. So, when we have to prove a goal $D \supset G$ at a world $w$, we have to move to each reachable world $w'$ in which $D$ holds. However, maybe nothing which holds at $w$ also holds at $w'$, so $G$ has to be proved in a completely new context where only $D$ holds.

4. THE FIRST-ORDER CASE

In this section we extend to the first-order case the modal interpretations defined in the previous section and we prove their correctness.

Of course no proof is needed for the language with open blocks and dynamic scoping. As said above, intuitionistic logic has been proved to be the underlying logic for such a language [14, 15, 26]. In [26] this result is proved for a language which also allows disjunctions in goals. By interpreting intuitionistic logic within modal logic S4 in the usual way, we can get a modal interpretation for the first-order language with open dynamic blocks, which is obviously correct.

In this section, we are mainly concerned with the language with open static blocks. We prove soundness and completeness of its operational semantics with respect to S4-Kripke semantics defined in Section 3.1. Completeness is proved by a Henkin-style canonical model construction, which is similar to the one given, for the case of intuitionistic logic programming, by Bonner, McCarty, and Vadaparty [4] and by Miller [28]. We also show that there is no loss of generality in restricting the S4-Kripke interpretations to those in which the domain at each world is the Herbrand universe. Similar results can be proved for the language with closed blocks.

Let us first consider open dynamic blocks, i.e., intuitionistic logic programming, in the first-order case. As done for the propositional case, to interpret such a language within modal logic S4, we apply the mapping from the language of intuitionistic logic to the language of S4 modal logic (see [13]). We have, in
addition, to take care of the quantifiers. The mapping (denoted by *) on the formulas of the first-order language \( L \), is the following:

\[
T^* = T, \\
A^* = \Box A \quad (A \text{ is an atomic formula}), \\
(\alpha \land \beta)^* = \alpha^* \land \beta^*, \\
(\alpha \implies \beta)^* = \Box (\alpha^* \implies \beta^*), \\
(\alpha \rightarrow \beta)^* = \Box (\alpha^* \supset \beta^*), \\
(\forall x \alpha)^* = \Box \forall x \alpha^*, \\
(\exists x \alpha)^* = \exists x \alpha^*.
\]

Notice that the modal operator \( \Box \) is put in front of the universal quantifier. This is needed when we refer to first-order intuitionistic Kripke models with monotonically increasing domains (also called nested domains). By applying the above mapping, we can extend the modal language \( L_1 \) defined in Section 3.1 to the first-order case as follows:

\[
L_1 \text{(first order):} \quad G := T \Box A \mid G_1 \land G_2 \exists x G \Box (D \supset G), \\
D := \Box (G \supset \Box A) \mid D_1 \land D_2 \mid \forall x D.
\]

It is clear that the modal translation \( L_2 \) of the language with open static blocks presented in Section 3.2 cannot be extended to the first-order case in the same way as for the intuitionistic language above. In fact, clauses in \( L_1 \) differ from those in \( L_2 \), since the former have an additional \( \Box \) in front of them. If in \( L_2 \) the modal operator \( \Box \) is put in front of universal quantifiers, and thus in front of clauses, the difference between \( L_1 \) and \( L_2 \) vanishes in the first-order case. Thus, for the language with open static blocks, the mapping * (defined in Section 3.2) is extended to quantified formulas in the following way:

\[
T^* = T, \\
A^* = \Box A \quad (A \text{ is an atomic formula}), \\
(\alpha \land \beta)^* = \alpha^* \land \beta^*, \\
(\alpha \implies \beta)^* = \Box (\alpha^* \implies \beta^*), \\
(\alpha \rightarrow \beta)^* = \alpha^* \supset \beta^*, \\
(\forall x \alpha)^* = \forall x \alpha^*, \\
(\exists x \alpha)^* = \exists x \alpha^*
\]

and the first-order modal interpretation of the language with open static blocks is as follows:

\[
L_2 \text{(first order):} \quad G := T \Box A \mid G_1 \land G_2 \exists x G \Box (D \supset G), \\
D := G \supset \Box A \mid D_1 \land D_2 \forall x D.
\]

So no modal operator is put in front of the universal quantifier in this case. From now on we will use \( L_1 \) and \( L_2 \) to refer to the corresponding first-order languages.
4.1. Correctness of the Modal Interpretation

In this subsection we aim at proving correctness for the modal interpretation of the language with open blocks with static scoping, i.e., at proving that, for all programs $P$ and goals $G$ of the language $L$,

$$P \vdash G \iff P^* \models G^*,$$

(2)

where $\vdash$ is operational derivability relation as defined in Section 2.2, $*$ is the mapping (1) defined above, and $\models$ is the satisfiability relation for the logic S4 defined in Section 3.1, where the Herbrand universe $U_L$ of the language $L$ is taken as the constant domain of each possible world.

An indirect proof of the above statement has been given in [19]. That proof makes use of some results presented in [17, 18], where a fixpoint and a model-theoretic semantics have been defined for the language $L$ with static blocks, with respect to which the operational semantics defined in Section 2.2 is sound and complete. In particular, the model-theoretic semantics is defined in terms of Kripke trees whose domain is the Herbrand universe for all possible worlds. In [19] we made use of this semantics, to prove the equivalence (2). Here, we will give a direct proof of it.

Let us now prove the soundness of the operational semantics with respect to the Kripke semantics above.

**Theorem 1 (Soundness).** Let $P$ be a program and $G$ be a closed goal in the language $L$. Then, for the language with open static blocks,

$$P \vdash G \implies P^* \models G^*.$$

**Proof.** In order to prove the thesis, we prove the following more general statement. For all programs $D_1, \ldots, D_n$,

$$D_1|\cdots|D_n \vdash G \implies P^* \models (D_1^* \supset (D_2^* \supset \cdots \supset (D_n^* \supset G^*) ))$$

This amounts to proving that if $D_1|\cdots|D_n \vdash G$, then for every interpretation $M = (W, R, e)$ and every world $w \in W$,

$$M, w \models P^* \models (D_1^* \supset (D_2^* \supset \cdots \supset (D_n^* \supset G^*) ))$$

The proof is by induction on the length $k$ of the derivation of $G$ from $D_1|\cdots|D_n$.

If $k = 0$, then it must be that $G = T$. In this case, the thesis holds trivially, since $T^* = T$ and, for all programs $D_1, \ldots, D_n$,

$$P^* \models (D_1^* \supset (D_2^* \supset \cdots \supset (D_n^* \supset T ) ))$$

We assume that the thesis holds for those goals whose operational derivation has length $j < k$ and we prove it for $k$, considering all possible cases for $G$:

- $G = T$. The thesis holds trivially.
- $G = A$. Assume that $D_1|\cdots|D_n \vdash A$, with a derivation whose length is $k$. Then for some $i$ ($1 \leq i \leq n$), there is a clause $G_1 \supset A \in [D_i]$ such that $D_1|\cdots|D_i \vdash G_1$ and $G_1$ has a derivation with length less than $k$. Hence, by inductive hypothesis,

$$P^* \models (D_1^* \supset (D_2^* \supset \cdots \supset (D_n^* \supset G_1^* ) ))$$
i.e., for every interpretation \( M = \langle W, R, e \rangle \) and every world \( w \in W \),
\[
M, w \models_s \Box (D^*_1 \supset \Box (D^*_2 \supset \cdots \supset \Box (D^*_i \supset G^*_i))).
\]
Hence, for each sequence of worlds \( w_1, \ldots, w_i \) such that
\[
w Rw_1 R \cdots R w_i \text{ and } M, w_j \models_s D^*_j \text{ for all } j = 1, \ldots, i,
\]
we have that
(a) \( M, w_i \models_s G^*_i \).

Since \( G_i \rightarrow A \in [D_i] \), \( G_i \rightarrow A \) is a ground instance of some clause in \( D_i \).
From \( M, w_i \models_s D^*_i \) and the fact that the domain of world \( w_i \) is \( U_L \), we can then conclude
(b) \( M, w_i \models_s G^*_i \supset \Box A \).

From (a) and (b), it follows that
\[
M, w_i \models_s \Box A
\]
and, by transitivity of \( R \), for all \( w_{i+1}, \ldots, w_n \) such that \( w_i Rw_{i+1} R \cdots R w_n \),
\[
M, w_n \models_s \Box A. \text{ Therefore, since } A^* = \Box A,
\]
\[
\models_s \Box (D^*_1 \supset \Box (D^*_2 \supset \cdots \supset \Box (D^*_n \supset A^*))).
\]

- \( G = G_1 \land G_2 \). Assume that \( D_1 \cdots |D_n| \vdash G_1 \land G_2 \), with a derivation whose length is \( k \). Then it must be that \( D_1 \cdots |D_n| \vdash G_1 \) and \( D_1 \cdots |D_n| \vdash G_2 \), with derivations of length less than \( k \). Hence, by inductive hypothesis,
\[
\models_s \Box (D^*_1 \supset \Box (D^*_2 \supset \cdots \supset \Box (D^*_n \supset G^*_1))).
\]
and
\[
\models_s \Box (D^*_1 \supset \Box (D^*_2 \supset \cdots \supset \Box (D^*_n \supset G^*_2))).
\]
From this, we can easily conclude that
\[
\models_s \Box (D^*_1 \supset \Box (D^*_2 \supset \cdots \supset \Box (D^*_n \supset (G^*_1 \land G^*_2)))).
\]

- \( G = D \Rightarrow G_1 \). Assume that \( D_1 \cdots |D_n| \vdash D \Rightarrow G_1 \), with a derivation whose length is \( k \). Then \( D_1 \cdots |D_n| \vdash G_1 \), with a derivations of length less than \( k \). Hence, by inductive hypothesis,
\[
\models_s \Box (D^*_1 \supset \Box (D^*_2 \supset \cdots \supset \Box (D^*_n \supset (D^* \supset G^*_1)))).
\]
which is precisely what we wanted to prove.

- \( G = \exists x G_1 \). Assume that \( D_1 \cdots |D_n| \vdash \exists x G_1 \), with a derivation whose length is \( k \). Then \( D_1 \cdots |D_n| \vdash G_1[t/x] \) for some closed term \( t \in U_L \), with a derivations of length less than \( k \). Hence, by inductive hypothesis,
\[
\models_s \Box (D^*_1 \supset \Box (D^*_2 \supset \cdots \supset \Box (D^*_n \supset G^*_1[t/x]))).
\]
Thus, for every interpretation \( M = \langle W, R, e \rangle \) and every world \( w \in W \),
\[
M, w \models_s \Box (D^*_1 \supset \Box (D^*_2 \supset \cdots \supset \Box (D^*_n \supset G^*_1[t/x]))).
\]
Hence, for each sequence of worlds \( w_1, \ldots, w_n \) such that
\[
wRw_1 \cdots Rw_n \quad \text{and} \quad M, w_j \models_{S4} D_j^* \quad \text{for all} \quad j = 1, \ldots, n,
\]
we have that \( M, w_n \models_{S4} G^*[t/x] \). Hence, \( M, w_n \models_{S4} \exists x G^*_1 \). Therefore,
\[
M, w \models_{S4} \Box (D_1^* \supset \Box (D_2^* \supset \cdots \supset \Box (D_n^* \supset \exists x G^*_1))).
\]

We will now prove completeness of the operational semantics with respect to the Kripke semantics above. Let us first state some lemmas which will be needed. These lemmas give some properties of the operational semantics for the language with static open blocks.

**Lemma 1 (Monotonicity).** Let \( \Gamma_1 \) and \( \Gamma_2 \) be (possibly empty) sequences of programs, let \( D \) be a program, and let \( G \) be a goal in the language \( L \). Then
\[
\Gamma_1 | D | \Gamma_2 \models G \quad \Rightarrow \quad \Gamma_1 | \Gamma_2 \models G.
\]

**Lemma 2.** Let \( \Gamma_1 \), \( \Gamma_2 \), and \( \Gamma_3 \) be (possibly empty) sequences of programs, let \( D \) be a program, and let \( G \) be a goal in the language \( L \). Assume that, for all clauses \( G' \rightarrow A \in [D] \),
\[
\Gamma_1 | \Gamma_2 \models G' \quad \Rightarrow \quad \Gamma_1 | \Gamma_2 \models A.
\]

Then
\[
\Gamma_1 | D | \Gamma_3 \models G \quad \Rightarrow \quad \Gamma_1 | \Gamma_2 | \Gamma_3 \models G.
\]

The proof of Lemma 2 can be found in Appendix A. The proof of Lemma 1 is similar to the proof of Lemma 2 and it is omitted.

We can now prove completeness of the operational semantics with respect to the Kripke semantics above. The completeness proof is given by constructing a canonical model for a given program \( P \), whose domain is constant and is the Herbrand universe of \( P, U_p \).

**Definition.** The canonical model for \( P \) is an S4-Kripke interpretation \( M^C = \langle W, R, e \rangle \), where:
- \( W = \{D_1| \cdots |D_n; n \geq 1 \text{ and all } D_i \text{ are programs in the language of } P\}; \)
- \( R \) is the reflexive and transitive closure of the binary relation on \( W \) containing all pairs \( (D_1| \cdots |D_n, D_1| \cdots |D_n| D) \), for all programs \( D_1, \ldots, D_n, D \) in the language of \( P \);
- \( e(D_1| \cdots |D_n) = \{A: D_1| \cdots |D_n \models A \text{ and } A \text{ is a ground atom in the language of } P\}. \)

Note that, by Lemma 1, if \( D_1| \cdots |D_n \models A \), then \( D_1| \cdots |D_{n+1}| \cdots |D_{n+m} \models A \) for all ground atoms \( A \) and programs \( D_{n+1}, \ldots, D_{n+m} \). Hence, the canonical model has the property that
\[
A \in e(w) \quad \Rightarrow \quad A \in e(w'),
\]
for all ground atoms \( A \) and all worlds \( w' \) such that \( wRw' \) (i.e., if \( A \) holds at a world, then it holds at all reachable worlds).

In the following theorem and corollaries we will assume that all programs and goals contain the same symbols as \( P \).
Theorem 2. Let $D, D_1, \ldots, D_n$ be programs and let $G$ be a closed goal. Then, for the language with open static blocks,

(i) $M^C, D_1 \cdots |D_n \models_{S_4} G^* \iff D_1 \cdots |D_n \vdash G$;

(ii) $M^C, D_1 \cdots |D_n \models_{S_4} D^*$, for all clauses $D \in [D_n]$.

**Proof.** We prove (i) and (ii) simultaneously, by induction on the structure of $G$ and $D$. Let us first prove (i). We consider all possible cases for $G$.

- If $G = T$, obvious, since $T^* = T$, $M^C, D_1 \cdots |D_n \models_{S_4} T$, and $D_1 \cdots |D_n \vdash T$, for all programs $D_1, \ldots, D_n$.

- If $G = A$, then

  \[ M^C, D_1 \cdots |D_n \models_{S_4} A^* \]
  iff
  \[ M^C, D_1 \cdots |D_n \models_{S_4} G^* \land \Box A \]
  iff for each world $w' \in W$ such that $(D_1 |D_n | D_n) \models \Box A$, $A \in \epsilon(w')$
  iff $A \in \epsilon(D_1 |D_n | D_n) \land |D_n+m | \models A$, for all programs $D_n+1, \ldots, D_n+m$, $m \geq 0$
  iff $D_1 \cdots |D_n \models_{S_4} A$, for all programs $D_n+1, \ldots, D_n+m$, $m \geq 0$
  (by definition of $M^C$)
  iff $D_1 \cdots |D_n \vdash A$ (the if part by Lemma 1; the only if part by taking $m = 0$).

- If $G = G_1 \land G_2$, then

  \[ M^C, D_1 \cdots |D_n \models_{S_4} (G_1 \land G_2)^* \]
  iff
  \[ M^C, D_1 \cdots |D_n \models_{S_4} G_1^* \land G_2^* \]
  iff
  \[ M^C, D_1 \cdots |D_n \models_{S_4} G_1^* \mathrm{ and } M^C, D_1 \cdots |D_n \models_{S_4} G_2^* \]
  iff $D_1 \cdots |D_n \models_{S_4} G_1$ and $D_1 \cdots |D_n \vdash G_2$ (by inductive hypothesis)
  iff $D_1 \cdots |D_n \vdash G_1 \land G_2$.

- If $G = \exists x G_1$, then

  \[ M^C, D_1 \cdots |D_n \models_{S_4} (\exists x G_1)^* \]
  iff
  \[ M^C, D_1 \cdots |D_n \models_{S_4} \exists x G_1^* \]
  iff $M^C, D_1 \cdots |D_n \models_{S_4} G_1^*[t/x]$ for some $t \in U_p$ (by inductive hypothesis)
  iff $D_1 \cdots |D_n \vdash G_1[t/x]$ for some $t \in U_p$
  iff $D_1 \cdots |D_n \models_{S_4} \exists x G_1$.

- If $G = D \Rightarrow G_1$, we prove the two directions separately.

  (Only if) We assume that $M^C, D_1 | \cdots |D_n \models_{S_4} (D \Rightarrow G_1)^*$. That is, $M^C, D_1 | \cdots |D_n \models_{S_4} \Box (D^* \Rightarrow G_1^*)$, i.e., for each world $w' \in W$ s.t. $D_1 | \cdots |D_n | Dw', M^C, w' \models_{S_4} D^* \Rightarrow M^C, w' \models_{S_4} G_1^*$. i.e., for all programs $D_n+1, \ldots, D_n+m$ ($m \geq 0$),

  \[ M^C, D_1 \cdots |D_n |D_n+1 | \cdots |D_n+m \models_{S_4} D^* \Rightarrow M^C, D_1 \cdots |D_n |D_n+1 | \cdots |D_n+m \models_{S_4} G_1^* \]

  In particular, it holds that

  \[ M^C, D_1 \cdots |D_n |D \models_{S_4} D^* \Rightarrow M^C, D_1 \cdots |D_n |D \models_{S_4} G_1^* \]

  Since $D$ is a subformula of $G$, by inductive hypothesis, (ii) holds for $D$ and, hence,

  \[ M^C, D_1 \cdots |D_n |D \models_{S_4} D^* \] for all clauses $D' \in [D]$. 

Moreover, since $D \in \{D\}$,

$$M^C, D_1 \cdots D_n | D \models_{S_4} D^*.$$ 

Thus, we can conclude that $M^C, D_1 | D_n | D \models_{S_4} G^*_1$. Since $G_1$ is a subformula of $G$, by inductive hypothesis, (i) holds for $G_1$ and, hence $D_1 | D_n | D \models G_1$. Thus,

$$D_1 | D_n \models D \Rightarrow G_1.$$ 

(if) We assume that $D_1 | D_n \models \Rightarrow G_1$. Hence, it must be that $D_1 | D_n | D \models G_1$. We want to show that

$$M^C, D_1 | D_n \models \Rightarrow G_1 \Rightarrow \Rightarrow G_1^*,$$

i.e., that, for all programs $D_{n+1}, \ldots, D_{n+m}$ ($m \geq 0$),

$$M^C, D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models_{S_4} D^* \Rightarrow M^C, D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models_{S_4} G^*_1.$$

Let us assume that $M^C, D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models_{S_4} D^*$, that is, all the clauses in $D^*$ are satisfied in the canonical model, at the world $D_1 | \cdots | D_{n+1} | \cdots | D_{n+m}$. Hence, all their ground instances are satisfied too; that is, for all clauses $G' \Rightarrow A \in \{D\}$,

$$M^C, D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models_{S_4} G^* \Rightarrow M^C, D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models_{S_4} A^*$$

(where $A^* = \square A$). By inductive hypothesis, since both $G'$ and $A$ are subformulas of $G$, we have that

$$D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models G' \Rightarrow M^C, D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models_{S_4} G'^*$$

and

$$M^C, D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models_{S_4} A^* \Rightarrow D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models A.$$ 

Thus, for all clauses $G' \Rightarrow A \in \{D\}$,

$$D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models G' \Rightarrow D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models A.$$ 

From this and the fact that $D_1 | D_n | D \models G_1$, by applying Lemma 2, we get

$$D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models G_1.$$ 

Since $G_1$ is a subformula of $G$, by inductive hypothesis, (i) holds for $G_1$ and we have

$$M^C, D_1 | D_n | D_{n+1} | \cdots | D_{n+m} \models_{S_4} G_1^*.$$
Thus, we have proved that, for all programs $D_{n+1}, \ldots, D_{n+m}$ ($m \geq 0$),

$$M^C, D_1 \cdots |D_n| D_{n+1} \cdots |D_{n+m} \vdash_{S4} D^*$$

$$\Rightarrow$$

$$M^C, D_1 \cdots |D_n| D_{n+1} \cdots |D_{n+m} \vdash_{S4} G^*_1.$$

Let us now prove (ii).

- $D = G \rightarrow A$. If $M^C, D_1 \cdots |D_n \vdash_{S4} G^*$, then, since $G$ is a subformula of $D$, (i) holds for $G$ and, hence, $D_1 \cdots |D_n \vdash G$. Since $G \rightarrow A \in [D_n]$, we have that $D_1 \cdots |D_n \vdash A$. By Lemma 1, $D_1 \cdots |D_n| D_{n+1} \cdots |D_{n+m} \vdash_{S4} A$ for all programs $D_{n+1}, \ldots, D_{n+m}$ ($m \geq 0$). Again, since $A$ is a subformula of $D$, (i) holds for $A$ and, hence, $M^C, D_1 \cdots |D_n| D_{n+1} \cdots |D_{n+m} \vdash_{S4} A$ for all $D_{n+1}, \ldots, D_{n+m}$ ($m \geq 0$), and thus $M^C, D_1 \cdots |D_n \vdash_{S4} \Box A$, i.e., $M^C, D_1 \cdots |D_n \vdash_{S4} A^*$. Therefore, $M^C, D_1 \cdots |D_n \vdash (G \rightarrow A)^*$.\[\]

- $D = D_1 \wedge D_2$. If $D_1 \wedge D_2 \in [D_n]$, then $D_1 \in [D_n]$ and $D_2 \in [D_n]$. Since, by inductive hypothesis, (ii) holds for the subformulas of $D$,

$$M^C, D_1 \cdots |D_n \vdash_{S4} D_1^*$$

and

$$M^C, D_1 \cdots |D_n \vdash_{S4} D_2^*.$$\[\]

Hence, $M^C, D_1 \cdots |D_n \vdash_{S4} (D_1 \wedge D_2)^*$.\[\]

- $D = \forall x D$. If $\forall x D \in [D_n]$, then $D[t/x] \in [D_n]$ for all $t \in U_r$. Since, by inductive hypothesis, (ii) holds for the subformulas of $D$,

$$M^C, D_1 \cdots |D_n \vdash_{S4} D^*[t/x]$$

for all $t \in U_r$.\[\]

Hence, $M^C, D_1 \cdots |D_n \vdash_{S4} (\forall x D)^*$.\[\]

The following Corollaries 1 and 2 are immediate consequences of Theorem 2.

**Corollary 1.** Let $P$ be a program and let $G$ be a closed goal in the language $L$. Then, for the language with open static blocks,

$$M^C, P \vdash_{S4} G^* \iff P \vdash G.$$

**PROOF.** From Theorem 2, point (i), for $n = 1$.\[\]

**Corollary 2.** Let $P$ be a program in the language $L$. Then, for the language with open static blocks,

$$M^C, P \vdash_{S4} P^*.$$

**PROOF.** From Theorem 2(ii), it follows that $M^C, P \vdash_{S4} D^*$ for all clauses $D \in [P]$. From this the thesis holds, since $P \in [P]$.\[\]

We can now prove the following completeness result.

**Corollary 3 (Completeness).** Let $P$ be a program and let $G$ be a closed goal of $L$. Then, for the language with open static blocks,

$$P^* \supset G^* \Rightarrow P \vdash G.$$

**PROOF.** Let us assume that $P^* \supset G^*$. Then, for every Kripke interpretation $M = \langle W, R, e \rangle$, for every $w \in W$,

$$M, w \vdash_{S4} P^* \Rightarrow M, w \vdash_{S4} G^*.$$
In particular, this holds for the canonical model $M^C$ and for the world $P$:

$$M^C, P \models_{S4} P^* \Rightarrow M^C, P \models_{S4} G^*.$$ 

Since, by Corollary 2, $M^C, P \models_{S4} P^*$, we can conclude that $M^C, P \models_{S4} G^*$, and, by Corollary 1, that $P \models G$. □

We have proved that the modal interpretation of the language with open static blocks within the logic S4 is correct. It has to be noticed, however, that the modal interpretation we have provided is also correct if we take the logic K4 instead of S4; that is, for a program $P$ and a closed goal $G$,

$$P \models G \iff \vdash_{K4} P^* \supset G^*.$$  \tag{3}

As a difference with respect to S4, in K4-Kripke interpretations the accessibility relation $R$ is transitive, but not reflexive. A proof of the statement (3) can be easily provided. Indeed, completeness, i.e., the if part of (3), holds since any S4 interpretation is a K4 interpretation and, therefore, $\vdash_{K4} P^* \supset G^*$ implies $\vdash_{S4} P^* \supset G^*$, which, by Corollary 3, implies $P \models G$. Soundness, i.e., the only if part of (3), can be proved precisely in the same way as the soundness result with respect to S4 given in Theorem 1. In fact, the proof of Theorem 1 does not make any use of reflexivity of the accessibility relation $R$.

In a similar way, it is possible to show that the modal interpretation we have provided for the language with closed blocks is correct both for the logic S4 and for the logic K (in K-Kripke interpretations there are no conditions on the accessibility relation $R$). In the following, however, we will consider interpretations of the block languages in the logic S4.

4.2. Herbrand Domains

In Section 4.1 we proved soundness and completeness of the operational semantics with respect to S4-Kripke semantics, for the language with open static blocks. In doing this, we have considered Kripke interpretations having the Herbrand universe as the constant domain for each world, as defined in Section 3.1. We will now prove that, for the language $L_2$, given a program $P$, there is no loss of generality in restricting to the S4-Kripke interpretations on the Herbrand domain.

In order to make this proof simpler, we will assume that the language $L_2$ does not contain function symbols, but only constants. We will make use of S4-Kripke interpretations as defined in [12]. As a difference w.r.t. the Kripke semantics in Section 3.1, here we will not assume that the domain is constant. On the contrary, the domain can change from one world to another and can be different from the Herbrand universe. The only restriction on domains is that the domain of a world $w$ is contained in the domain of all worlds reachable from $w$, i.e., domains are increasing.

Let $L_M$ be a first-order modal language containing countably many constants, variables, and predicate symbols. Let $L_M^*$ be the language obtained by adding to $L_M$ countably many new constants. We consider Kripke interpretations which have the constants of $L_M^*$ as domain and which interpret each constant as naming itself.
Thus, in the following text, we only need to specify the interpretation for predicate symbols and not for constants.

**Definition.** An S4-Kripke interpretation is an ordered quadruple $M = \langle W, R, \mathcal{D}, e \rangle$, where:

- $W$ is a nonempty set of worlds;
- $R$ is a binary relation on $W$ (accessibility relation) which is reflexive and transitive;
- $\mathcal{D}$ is a function from $W$ to nonempty sets of constants $L^*_M$ (it associates a domain with each world), satisfying the condition for $w, w' \in W$, if $wRw'$ then $\mathcal{D}(w) \subseteq \mathcal{D}(w');$
- $e$ is a function assigning, to each world $w \in W$, a set $e(w)$ of ground atoms $p(a_1, \ldots, a_n)$, where each $a_i$ is a constant in $\mathcal{D}(w)$.

Let $\models_{S_4}$ be a relation between members of $W$ and statements $L^*_M$ ($M, w \models_{S_4} X$ means: $X$ is true at $w$ in the interpretation $M$) satisfying, for all $w \in W$, the following conditions:

- $M, w \models_{S_4} T$
- $M, w \models_{S_4} p(a_1, \ldots, a_n) \text{ iff } p(a_1, \ldots, a_n) \in e(w)$
- $M, w \models_{S_4} \alpha \land \beta \text{ iff } M, w \models_{S_4} \alpha \text{ and } M, w \models_{S_4} \beta$
- $M, w \models_{S_4} \alpha \lor \beta \text{ iff } M, w \not\models_{S_4} \alpha \text{ or } M, w \models_{S_4} \beta$
- $M, w \models_{S_4} \forall x \alpha \text{ iff for each } c \in \mathcal{D}(w), M, w \models_{S_4} \alpha[c/x]$
- $M, w \models_{S_4} \exists x \alpha \text{ iff for some } c \in \mathcal{D}(w), M, w \models_{S_4} \alpha[c/x]$
- $M, w \models_{S_4} \Box \alpha \text{ iff for all world } w' \in W \text{ such that } wRw', M, w' \models_{S_4} \alpha.$

A closed formula $\alpha$ of the language $L^*_M$ is $S_4$-satisfiable if there is a S4-Kripke interpretation $M = \langle W, R, \mathcal{D}, e \rangle$ and some $w \in W$ with every constant of $\alpha$ in $\mathcal{D}(w)$ such that $M, w \models_{S_4} \alpha$. We say that $\alpha$ is a $S_4$-valid formula ($\models_{S_4} \alpha$) if, for every S4-Kripke interpretation $M = \langle W, R, \mathcal{D}, e \rangle$, for every $w \in W$ with every constant of $\alpha$ in $\mathcal{D}(w)$, $M, w \models_{S_4} \alpha$.

Notice that, as a difference with respect to the definition in Section 3.1, when dealing with the first-order case, it is meaningful to speak of the truth of a formula in an interpretation at a certain world only if the constants in that formula are in the domain of the world. When function symbols are present, there are additional problems, since each function symbol can be given a different interpretation at each different world. For simplicity, we have only considered the case when the language contains no function symbols. For a survey of the different systems for quantified modal logic, see [16].

Given a program $P$ in the modal language $L^*_2$, we want to show that, without loss of generality, we can take into account only S4 interpretations with constant domain $U_p$; that is, interpretations (as those defined Section 3.1) in which, for all worlds $w \in W$, $\mathcal{D}(w) = U_p$. In essence, for this fragment of S4, it is not necessary to consider interpretations containing the additional constants in $L^*_2$. We can prove the following proposition.
Proposition 1. Let \( P \) and \( G \) be a program and a closed goal of the modal language \( L_2 \). If, for a given S4 interpretation \( M = \langle W, R, \mathcal{D}, e \rangle \) and a world \( w \in W \) with every constant of \( P \) in \( \mathcal{D}(w) \), \( M, w \models_{S4} P \land \neg G \), then there is an S4 interpretation \( M_H = \langle W, R, \mathcal{D}_H, e_H \rangle \) with constant domain \( U_p \), such that \( M_H, w \models_{S4} P \land \neg G \).

PROOF. See Appendix B. \( \square \)

In the proposition above we have assumed that the goal \( G \) contains the same symbols as \( P \). It is well known that such a proposition does not hold if we take, instead of \( D \land \neg G \), an arbitrary modal formula \( \alpha \) of the language \( L_2 \). Consider the formula \( \alpha = p(a) \land \neg BF \), where \( BF = \forall x \Box p(x) \lor \neg \forall x p(x) \), which is an instance of the Barcan formula. Then \( U_a = \{a\} \). Let us assume that \( b \) is an additional constant in \( L_2 \). Let \( M \) be the interpretation \( \langle \{w_1, w_2\}, R, \mathcal{D}, e \rangle \), such that \( R = \{(w_1, w_2), (w_1, w_1), (w_2, w_2)\} \), \( a \in \mathcal{D}(w_1) \), \( b \notin \mathcal{D}(w_1) \), \( a, b \in \mathcal{D}(w_2) \), and \( e(w_1) = e(w_2) = \{p(a)\} \). So \( p(a) \) is true both at \( w_1 \) and at \( w_2 \) while \( p(b) \) is false at \( w_2 \); \( p(b) \) has no truth value in \( w_1 \) since \( b \) is not in the domain of \( w_1 \). It is clear that \( BF \) is not true at \( w_1 \) in \( M \) and thus \( M, w_1 \models_{S4} \neg BF \). Therefore, \( M, w_1 \models_{S4} p(a) \land \neg BF \), i.e., \( M, w_1 \models_{S4} \alpha \). On the contrary, there is no Kripke interpretation with constant domains in which \( \neg BF \) is true at some world. In fact, \( BF \) is true at any world in any interpretation with constant domain. In particular, we cannot find a model \( M_H \) with constant domain \( U_a = \{a\} \) and a world \( w' \) such that \( M_H, w' \models_{S4} p(a) \land \neg BF \). Thus the proposition above does not hold for the formula \( \alpha = p(a) \land \neg BF \).

By Proposition 1, we can consider, for the language \( L_2 \), only Kripke interpretations having as constant domain the Herbrand universe, as usual in logic programming. In fact, the following corollary holds.

Corollary 4. For all programs \( P \) and goals \( G \) of \( L_2 \), \( \models_{S4} P \supset G \) iff \( P \supset G \) is true in all Kripke S4-interpretations with constant domain \( U_p \) (written \( \models_{S4,H} P \supset G \)).

PROOF. (Only if) It is obvious that if \( P \supset G \) is true in all Kripke S4 interpretations, then \( P \supset G \) is true in all Kripke S4 interpretations with constant domain \( U_p \).

(If) We make use of Proposition 1 to prove that if \( P \supset G \) is true in all Kripke S4 interpretations with constant domain \( U_p \), then \( \models_{S4} P \supset G \). In fact, let us start from the hypothesis \( \models_{S4,H} P \supset G \). If \( P \supset G \) is not S4-valid, then its negation \( P \land \neg G \) is true in some model \( M \) at some world \( w \). Therefore, by Proposition 1, there is an interpretation \( M_H \) with constant domain \( U_p \), such that \( M_H, w \models_{S4} P \land \neg G \). Hence, \( P \supset G \) is not true in all Kripke S4-interpretations with constant domain \( U_p \), which contradicts the hypothesis \( \models_{S4,H} P \supset G \). \( \square \)

A similar result can also be obtained for the language \( L_1 \) with open dynamic blocks and for the language \( L_3 \) with closed blocks. Therefore, for all these languages, we are allowed to consider only Kripke interpretations with the Herbrand universe as constant domain. Proposition 2 in Appendix B is the analogue, for the language \( L_1 \), of Proposition 1 above.

Notice that, when Kripke interpretations with constant domain \( U_p \) are considered for the language \( L_1 \), the modal operator \( \Box \) in front of universal quantifiers can be eliminated. It is easy to see that, by replacing \( \Box \forall x D \) with \( \forall x D \) in the first-order modal language \( L_1 \), an equivalent language is obtained (see Proposition 3 in Appendix B). In the following, therefore, by \( L_1 \) we will refer to the language without \( \Box \) in front of \( \forall x D \).
5. AN INTEGRATED LANGUAGE

We have shown that the different languages with blocks considered in the paper all can be interpreted within modal logic S4. Hence, it is possible to integrate them by defining a single modal language which allows different kinds of scope rules for clause definitions, as follows:

\[ L_4: \quad G := T \| A | G_1 \land G_2 | \exists x G | \square (G \supset G), \]

\[ D := \square (G \supset \square A) | G \supset \square A | D_1 \land D_2 | \forall x D. \]

Note that since in this language different kinds of clause definitions are allowed (also in the same block), it is no longer meaningful to speak of static or dynamic blocks. Instead, we will speak of static clause definitions [i.e., definitions of the form \( G \supset \square A \)] and dynamic clause definitions [of the form \( \square (G \supset \square A) \)]. With this language it is feasible to distinguish among the static and the dynamic parts of a program, thus allowing a partial use of compilation techniques when static clauses are employed.

To define the operational semantics of this language, since static clause definitions are allowed, it is necessary to consider the derivability of a goal \( G \) from a list of programs \( D_1 | \cdots | D_n \), as in the case of open static blocks. The derivability of a closed goal \( G \) from a list of programs \( D_1 | \cdots | D_n \) can be defined by induction on the structure of \( G \) as follows:

- \( D_1 | \cdots | D_n \vdash T; \)
- \( D_1 | \cdots | D_n \vdash \square A \) if, for some \( i, 1 \leq i \leq n, \)
  there is a clause \( \square (G \supset \square A) \in [D_i] \) and \( D_1 | \cdots | D_n \vdash G \)
  or there is a clause \( G \supset \square A \in [D_i] \) and \( D_1 | \cdots | D_n \vdash G; \)
- \( D_1 | \cdots | D_n \vdash G_1 \land G_2 \) if \( D_1 | \cdots | D_n \vdash G_1 \) and \( D_1 | \cdots | D_n \vdash G_2 \);
- \( D_1 | \cdots | D_n \vdash \exists x G \) if there is a closed term \( t \) such that \( D_1 | \cdots | D_n \vdash G[t/x] \);
- \( D_1 | \cdots | D_n \vdash \square (D \supset G) \) if \( D_1 | \cdots | D_n \vdash D \vdash G. \)

The operational semantics of the implication goal \( \square (D \supset G) \) is the same as for open static blocks (the set of local declarations \( D \) is added to the list of programs as the tail element), but a different use of the list of programs is done each time a new clause is selected, according to its kind. Notice that when a static clause \( G \supset \square A \) is selected from \( D_i \) to prove an atomic goal \( \square A \), then the clauses in \( D_{i+1} | \cdots | D_n \) cannot be used any more to prove \( G \), since the clauses of inner blocks are not visible from external ones (and thus from \( G \)). When a dynamic clause \( \square (G \supset \square A) \) is selected in \( D_i \) to prove \( \square A \), then all the programs \( D_1, \ldots, D_n \) in the list can be used to prove \( G \) (i.e., the clauses in the different blocks are regarded as being indistinguishable).

When a static clause is used in a program, its body can be considered as completely defined in the enclosing blocks and no successive update can affect it. If the body is true, then the head of the clause will be visible in the nested blocks. On the other hand, the body of a dynamic clause also can be proved dynamically using clauses introduced by updates.

It has to be noticed that clauses of the form \( G \supset A \) (those occurring in the language \( L_3 \)) cannot be easily added to the language \( L_4 \). Moreover, their addition to \( L_4 \) would not allow closed blocks to be defined. In fact, closed blocks are
provided by the language \( L_3 \) by only allowing the modal operator to occur in front of implications in goals. To recover the possibility of defining closed blocks, it suffices to introduce in the language \( L_4 \) a new universal modal operator \([c]l\) (we move to a multimodal logic) for which the axioms of S4 modal logic hold. Consider an extended language with the following syntax:

\[
L_5: \quad G := T|\Box A|G_1 \land G_2|\exists x G|\Box (D \supset G)\langle [c]l\rangle (D \supset G),
\]

\[
D ::= (G \supset \Box A)|G \supset \Box A|D_1 \land D_2|\forall x D
\]

A Kripke interpretation for this language has to contain a second accessibility relation associated with the additional modal operator \([c]l\). Since the modal operator \([c]l\) only occurs in front of implication goals, its effect consists of closing a context, i.e., to allow closed blocks. In fact, to deal with such closed implications the following new rule has to be added to the operational semantics for the language \( L_4 \):

- \( D_1|\cdots|D_n \mid [c]l\langle D \supset G \rangle \text{ if } D \vdash G \).

When a closed implication \([c]l\langle D \supset G \rangle \) has to be proved, the current context (list of programs) is removed and \( G \) is proved in the local set of clauses \( D \).

In the rest of this section we will introduce some syntactic sugar on this integrated language in order to explore its potential uses in building structured logic programs. Though the language defined above is probably too complex to be used in practice, we believe that, on its base, a higher level language subsequently could be defined which allows for more concise notations (for instance, to distinguish static clauses from dynamic ones and open blocks from closed ones) and also for a more constrained combination of the different constructs.

### 5.1. Some Syntactic Sugar to Define Modules

As already mentioned in Section 2, modules can be defined in a language with blocks if module names are associated with sets of clauses and used in implication goals. For instance, a module \( m \) can be defined as consisting of the set of clauses \( D \) by the definition

\[ m \text{ is-mod } D, \]

where \( m \) is a term, the name of the module, and the language of the clauses in \( D \) is the modal language \( L_5 \) or one subset of it. Once the name of the module has been associated with a set of clauses in this way, then the module name can be used in implication goals. We can write

\[ \Box (m \supset G) \]

to say that the goal \( G \) has to be proved in the module \( m \). Since in implication goals \( \Box (D \supset G) \) the clauses \( D \) and the goal \( G \) are allowed to share variables, in a module definition, \( m \text{ is mod } D, \) \( D \) is allowed to contain free variables and \( m \) must be parametric in these variables.

The operational semantics of this language with modules is the same as for the block language if each occurrence of a module name is replaced with the corresponding set of clauses. Of course, this preprocessing can be performed only if the modules are not recursively defined. The different kinds of clauses that a module
contains cause how the module interacts with other modules. In the following text we will consider several kinds of modules and ways to combine them.

In a very simplistic view, we can assume that a module is a closed environment which \textit{exports} every predicate symbol defined inside it and \textit{imports} a predicate symbol from other modules by explicitly referring to those modules. We can obtain closed modules of this kind, by defining each module $m_1, \ldots, m_k$ as a set of clauses (either static or dynamic) in the language $L_5$. The goal $[cl](m_i \supset G)$ succeeds if $G$ succeeds from the clauses in module $m_i$. Such implication goals, of course, can occur within each module. Thus a module $m_j$ is able to import a predicate $A$ from $m_i$ explicitly by the implication goal $[cl](m_i \supset A)$. Moreover, every proposition defined in a module can be regarded as exported; that is, it can be queried from other modules. Notice that in order to obtain closed modules, it is essential to use the modal operator $[cl]$ in front of implication goals.

A more complex kind of open module (called \textit{unit}) has been defined in [29]. The operational semantics of this language with units is quite similar to the one for static open blocks. A difference is that \textit{predicate overriding} is employed; that is, the most recent definition of a predicate in a sequence overrides the previous ones. To model this kind of module in our language (apart from predicate overriding) is straightforward: it suffices to allow each module to contain only (static) clauses of the form $G \supset \square A$, as in language $L_2$, where $G$ also may contain implication goals of the form $\square (m_j \supset G)$ corresponding to the "context extension formulas" $m \supset G$ in [29]. In this way, modules are no longer closed and can be composed. It has to be noticed, however, that the language in [29] also allows for the definition of mutually recursive units that are not allowed here.

An example of module composition, which is adapted from [29] (we will not introduce quantifiers explicitly and we will assume that all clauses are implicitly universally quantified), is

\begin{verbatim}
authors is-mod \{\square wrote(Person, Something) \supset \square author(Person)\}
books is-mod \{T \supset \square wrote(plato, republic). \}

T \supset \square wrote(homer, iliad).
\square (authors \supset \square author(Person)) \supset \square writer(Person),),
\end{verbatim}

where the goal \(\square (books \supset \square writer(plato))\) has the derivation

\begin{verbatim}
\vdash \square (books \supset \square writer(plato))
books \vdash \square writer(plato)
books \vdash \square (authors \supset \square author(plato))
books\|authors \vdash \square author(plato)
books\|authors \vdash \square wrote(plato, Something)
books \vdash T.
\end{verbatim}

Here, according to the static visibility rules, the inner module authors implicitly imports all the facts that are provable in the module books, but not vice versa.
Note that modules can also be composed by nesting them in the initial goal. For instance, given a module sort defining a predicate quicksort

```plaintext
sort is-mod { ...
      □ greater_than(X, Y) ... ⊃ □ quicksort(...). 
      .... } 
```

and two different modules defining a predicate greater_than,

```plaintext
integers is-mod { ...
     ............ ⊃ □ greater_than(X, Y). 
     .... } 
```

```plaintext
char is-mod { ...
     ............ ⊃ □ greater_than(X, Y), 
     .... } 
```

the two goals

```plaintext
□ (integers ⊃ □ (sort ⊃ □ quicksort(...)))
```

and

```plaintext
□ (char ⊃ □ (sort ⊃ □ quicksort(...)))
```

will compute quicksort in two different environments.

In a similar way, in this language it is possible to define a module system in which the composition of modules consists of the union of the modules. In this case, all clauses contained in each module must be dynamic, i.e., of the form □ (G ⊃ □ A), so that, given the query □ (m₁ ⊃ □ (mᵢ ⊃ G)), the goal G is proved in the union of the modules mᵢ and m. Of course, these modules are those defined in [26] by adding some syntactic sugar to the language with dynamic blocks.

The block language we have used to build the different kinds of modules is the language \( L_s \). Therefore, in the same program it is also possible to define different kinds of modules and even to define a module containing both static and dynamic clauses. This provides a considerable flexibility, since it allows us to make use of dynamic features only when needed, while leaving the rest of the program static.

The presence of static and dynamic clause definitions in a module allows us to specify which clauses of the module are exported to other modules. Thus, if a module \( m \) contains a dynamic clause of the form □ (G ⊃ □ A), such a clause is intended to be visible to other modules. Of course, since modules are composed by means of implication goals as

```plaintext
□ (m₁ ⊃ □ (m₂ ⊃ □ (m₃ ⊃ G))),
```

the direction of the export is determined by the nesting of implication goals: module \( m₁ \) exports toward all the more deeply nested modules (\( m₂ \) and \( m₃ \)), while \( m₂ \) only exports to \( m₃ \) (and not to \( m₁ \)). So, each dynamic clause defined in a module is visible to internally nested modules or, in other words, can be affected by successive updates.
On the contrary, a clause of the form \( G \supset \Box A \) in the module \( m \) is not visible to internally nested modules. The possibility of proving \( G \) is not affected by successive updates. Consider, for instance, the following example.

**Example 5.1.** If we have the two modules

\[
m_4 \text{ is-mod } \{ \Box d \supset \Box a \\
\quad \Box(\Box b \supset \Box d) \}
\]

\[
m_5 \text{ is-mod } \{ T \supset \Box b \},
\]

then the goal \( \Box(m_4 \supset \Box(m_5 \supset \Box d)) \) succeeds from the program consisting of the two modules (since the body \( \Box b \) of the second clause of \( m_4 \) can be proved in the module \( m_5 \)), while the goal \( \Box(m_4 \supset \Box(m_3 \supset \Box a)) \) fails, since the first clause in \( m_4 \) is static and the proof of its body \( \Box d \) cannot make use of clauses defined in more internal modules as \( m_5 \). Notice that not only \( \Box d \) has to be resolved with a clause in \( m_4 \), but all the proof of \( \Box d \) has to be done in \( m_4 \).

### 5.2. Statically Configured Module Systems

In [6] a distinction is made between *statically* and *dynamically configured* module systems. A statically configured system is defined as a system where hierarchies among units (i.e., modules) are specified when units are defined. In these systems the context in which a unit is used does not depend on the dynamic sequence of goals, but is always fixed. On the contrary, in a dynamically configured system the context in which a module is used can be different in different queries (see the above sort example). To define statically configured units, in [6], the statement

\[
\text{unit}(m_4, \text{static}([m_2, m_3, m_4]))
\]

is introduced, whose meaning is that whenever the unit \( m_4 \) is used, it is used in the context of the modules \( m_2, m_3, m_4 \).

In our language, statically configured modules can be allowed by regarding the above static unit definition as syntactic sugar. Its meaning is that each occurrence of the implication goal \( \Box(m_1 \supset G) \) in the program has to be replaced with the implication goal

\[
[cl](m_4 \supset \Box(m_3 \supset \Box(m_2 \supset \Box(m_1 \supset G))))).
\]

In this way, the context in which \( m_4 \) is used is always the closed context containing \( m_2, m_3, \) and \( m_4 \). A preprocessing step is needed to make the replacement above and, in general, more steps are needed if more than one unit is defined as statically configured.

In [6] it was shown that the choice of statically configured modules with dynamic visibility rules for clauses is suitable for dealing with inheritance based systems. Indeed, statically configured modules can be used to represent the static dependencies among modules in a hierarchy. We rephrase in our language an example presented in [6] and taken from [22]. Rephrasing is needed since overriding is not provided by our language. Notice also that we do not deal with multiple inheritance.
Example 5.2. Let us consider three modules, named, respectively, \textit{animal}, \textit{bird}, and \textit{tweety}. Since what is true for animals is also true for birds, the \textit{bird} module inherits from the \textit{animal} module. Moreover, the module \textit{tweety} inherits from \textit{bird} and thus from \textit{animal}. Let us assume that there are no modules more general than \textit{animal}. To model this situation, the following static unit declarations have to be introduced:

\begin{verbatim}
unit(animal, static([])).
unit(bird, static([animal])).
unit(tweety, static([bird])).
\end{verbatim}

The modules \textit{animal}, \textit{bird}, and \textit{tweety} are defined as follows:

\begin{verbatim}
animal is-mod
\{ T \supset \Box \text{mode}(walk).
\Box(\Box \text{no_of_legs}(2) \supset \Box \text{mode}(run)).
\Box(\Box \text{no_of_legs}(4) \supset \Box \text{mode}(gallop)).\}
\end{verbatim}

\begin{verbatim}
bird is-mod
\{ T \supset \Box \text{no_of_legs}(2).
T \supset \Box \text{covering}(feather).\}
\end{verbatim}

\begin{verbatim}
tweety is-mod
\{ T \supset \Box \text{owner}(fred).\}
\end{verbatim}

Because of the static configuration, when a goal \( \Box(\Box \text{tweety} \supset \Box \text{mode}(run)) \) is called, it is replaced, by a preprocessing, by the goal

\begin{verbatim}
[cl](animal \supset \Box(\text{bird} \supset \Box(\text{tweety} \supset \Box \text{mode}(run))))
\end{verbatim}

So, the goal \( \Box \text{mode}(run) \) is proved in the list of modules, \textit{animal/bird/tweety}. Since the clause for \( \Box \text{mode}(run) \) in the module \textit{animal} is dynamic, the subgoal \( \Box \text{no_of_legs}(2) \) can be proved in the nested (more specific) module \textit{bird} and, therefore, the call succeeds. Thus, the use of dynamic clauses provides something similar to the \textit{self} annotation in object-oriented languages. On the contrary, the use of static clauses has similarities with the \textit{super} annotation: the definition of predicates in the body of a static clause is looked for in less specific modules (or in the current one).

5.2. Combining Block and Module Constructs

We have seen that, when defining a module as a set of clauses, the kind of clauses in the module (whether they are dynamic or static) determine how the module interacts with other modules. In any case, the clauses in a module \( m_i \) can contain implication goals both of the form \( \Box (m_j \supset G) \), which calls for a proof of the goal \( G \) in the module \( m_j \), and of the form \( \Box (D \supset G) \), that is a block. Therefore, the internal language of a module can be a language with blocks.
The possibility of combining blocks and modules allows us, for instance, to define closed modules (by means of the operator [cl]) whose internal language is a language with blocks, open static blocks as well as dynamic blocks. This seems to be an interesting possibility, since we believe that while open blocks are well suited for programming in the small, they do not always well fit the needs of programming in the large. A notion of module as a closed environment, with a rather limited and disciplined interaction with the external environment (to be specified by an interface) seems to be more adequate in this case.

6. CONCLUSIONS AND RELATED WORK

In this paper we have proposed a use of modal logic to introduce structuring constructs in logic programming. In particular, we have provided a modal reconstruction of several logic languages with embedded implications (blocks). The languages in [14, 25, 23] that have dynamic visibility rules for locally defined clauses are based on intuitionistic logic and have a straightforward interpretation within modal logic S4. We have defined a similar modal interpretation for languages with static or lexical visibility rules [18, 30] and for a language with closed blocks [18] in which closed blocks are the object-level analogue of the demo operator introduced by Bowen and Kowalski [5]. Some of the results of this paper were first presented in a shorter paper [19].

We have shown that an integrated language with blocks can be defined within this modal framework and it is suitable to provide different kinds of module constructs. It can provide a notion of closed module along with the possibility of dynamically composing modules as in [26] and also of having a more static composition of modules as in [29]. Moreover, to represent static dependencies of modules in a hierarchy, the definition of static configurations of modules [6] is also allowed. In essence, modal logic can provide a unifying framework in which different proposals for blocks and modules in logic programming can be captured so that different structuring constructs can be integrated in the same language and also used in the same program.

An alternative way to define modules was proposed in [3]. There, instead of using syntactic sugar to assign a name to each module, a multimodal logic is used; each modal operator refers to a module.

A similar use of multimodal logic to structure logic programs has been done in [11], where a language with modules inspired by the proposal by Monteiro and Porto [29] is defined. To design a module, a set of predicate names is used (the predicate names whose definitions are contained in the module) and this allows predicate overriding to be modelled.

In [27], to allow lexical scoping in the context of intuitionistic logic programming, a use of universal quantification in goals and clause bodies has been proposed. A goal \( \forall x G \), where the variable \( x \) can range over individuals, functions, or predicates, succeeds from a program \( P \), if \( G \) succeeds from \( P \) when \( x \) is replaced in \( G \) by a new constant \( c \). Universal quantification can provide lexical scoping for individual, function, and predicate constants. In particular, the use of predicate variables allows static visibility rules for clauses to be defined: given the goal \( \forall p(D \Rightarrow G) \), the predicate variable \( p \) is bound to a predicate constant which is only visible in \( D \) and \( G \). In general, such use of universal quantification allows abstract data types to be defined.
A different framework for structuring logic programs has been defined in [7]. There a distinction is made between conservative and evolving policies and between statically and dynamically configured systems. We have already discussed the distinction between statically and dynamically configured systems in the previous section and we have shown that static configurations of modules can be defined by making use of the modal operator \([c]\) for closed modules. The distinction between conservative and evolving policies roughly corresponds to our distinction between static and dynamic visibility rules. As a difference, while we regard clauses as being either static or dynamic, in [7] such a behavior is referred to the goals in clause bodies.

As pointed out in the Introduction, the aim of this paper is not to define a general purpose modal extension of logic programming, but to show how a very limited modal extension of Horn clause logic can provide different structuring facilities and can allow them to be integrated in a single language. More general modal extensions are considered in [1], where resolution proof systems for several modal logics are presented, and in [10], where an extension of Prolog with modal operators, called MOLOG, is presented. In [10] a resolution procedure, close to Prolog resolution, is defined for modal Horn clauses in the logic S5 which contains only universal modal operators of the form \(\text{Know}(a)\).

A temporal logic programming language TEMPLOG, which allows certain temporal operators in Horn clauses, has been defined in [2]. In TEMPLOG a distinction is made between initial clauses and permanent clauses, which appears to be similar to our distinction between local and dynamic clauses. No embedded implications are allowed in TEMPLOG.

In [33], a modal operator \textit{assume} is proposed, which can replace many uses of \textit{assert} in a logic program. The operator \textit{assume} allows the program to be updated during the computation by addition of new facts (atomic formulas).

To conclude, we want to mention briefly some other approaches that have been proposed to introduce module constructs in logic programming: the algebraic approach proposed in [32] and also developed in [21], the approach based on second order logic [31, 9], and, finally, the metalevel approach [5, 8].

We believe that our proposal has the main advantage of accommodating different structuring facilities in a unique framework and allowing them to be combined in an integrated language, for which an operational semantics can be defined in the same style as for ordinary Horn clause logic and whose model theoretic semantics enjoys very similar properties (for instance, as for Horn clause logic, Herbrand domains are sufficient).

\section*{APPENDIX A}

The following repetition of Lemma 2 states a property of the operational semantics for the language with static open blocks.

\textbf{Lemma 2.} Let \(\Gamma_1, \Gamma_2, \) and \(\Gamma_3\) be (possibly empty) sequences of programs, let \(D\) be a program, and let \(G\) be a goal in the language \(L\). Assume that, for all clauses \(G' \rightarrow A \in [D]\),

\[\Gamma_1, \Gamma_2 \vdash G' \Rightarrow \Gamma_1, \Gamma_2 \vdash A.\]
Then
\[ \Gamma_1 | D \Gamma_2 \vdash G \Rightarrow \Gamma_1 | \Gamma_2 \Gamma_3 \vdash G. \]

**Proof.** The proof is by induction on the length \( h \) of the derivation of \( G \) from \( \Gamma_1 | D \Gamma_3 \).

If \( h = 0 \), then it must be that \( G = T \), and the thesis holds trivially. We assume that the thesis holds for those goals whose operational derivation has length \( j < h \) and we prove it for \( h \), considering all possible cases for \( G \).

- \( G = T \). The thesis holds trivially, as above.
- \( G = A \). Assume that \( D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash A \), with a derivation whose length is \( h \). Then, there are three cases:
  1. There is a clause \( G' \rightarrow A \in [D] \) such that \( D_1 \cdots | D_i \vdash G' \), for some \( i = 1, \ldots, k \).
  2. There is a clause \( G' \rightarrow A \in [D] \) such that \( D_1 \cdots | D_k \Gamma_2 \vdash G' \).
  3. There is a clause \( G' \rightarrow A \in [D] \) such that \( D_1 \cdots | D_k | D_{k+1} \cdots | D_i \vdash G' \), for some \( i = k + 1, \ldots, n \).

In case (1), it trivially holds that \( D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash A \).

In case (2), since \( G' \) has a derivation with length less than \( h \), by inductive hypothesis, \( D_1 \cdots | D_k \Gamma_2 \vdash G' \). Moreover, given that \( G' \rightarrow A \in [D] \) and the hypothesis (i), from \( D_1 \cdots | D_k \Gamma_2 \vdash G' \) we can conclude that \( D_1 \cdots | D_k \Gamma_2 \vdash A \) and, hence, by monotonicity (Lemma 1), that \( D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash A \).

In case (3), since \( G' \) has a derivation with length less than \( h \), by inductive hypothesis, \( D_1 \cdots | D_k | D_{k+1} \cdots | D_i \vdash G' \). Since \( G' \rightarrow A \in [D] \), then \( D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash G \).

- \( G = G_1 \wedge G_2 \). Assume that \( D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash G_1 \wedge G_2 \), with a derivation whose length is \( h \). Then it must be that \( D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash G_1 \) and \( D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash G_2 \), with derivations of length less than \( h \). Hence, by inductive hypothesis,

\[
D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash G_1,
\]
\[
D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash G_2,
\]

and, therefore,
\[
D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash G_1 \wedge G_2.
\]

- \( G = D \Rightarrow G_1 \). Assume that \( D_1 \cdots | D_k | D_{k+1} \cdots | D_n \vdash D \Rightarrow G_1 \), with a derivation whose length is \( h \). Then \( D_1 \cdots | D_k | D_{k+1} \cdots | D_n | D_1 \vdash G_1 \), with a derivation of length less than \( h \). Hence, by inductive hypothesis,

\[
D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n | D_1 \vdash G_1,
\]

and, thus,
\[
D_1 \cdots | D_k \Gamma_2 | D_{k+1} \cdots | D_n \vdash D_1 \Rightarrow G_1.
\]

- \( G = \exists x G_1 \). Assume that \( D_1 \cdots | D_k | D_{k+1} \cdots | D_n \vdash \exists x G_1 \), with a derivation whose length is \( h \). Then it must be that \( D_1 \cdots | D_k | D_{k+1} \cdots | D_n \vdash G_1[t/x] \)
for some closed term $t$, with a derivation of length less than $h$. Hence, by
inductive hypothesis,

$$D_1 \cdots |D_k| \Gamma_2|D_{k+1}| \cdots |D_n \vdash G_1[t/x]$$

for some closed term $t$. Therefore,

$$D_1 \cdots |D_k| \Gamma_2|D_{k+1}| \cdots |D_n \vdash \exists x G_1. \quad \Box$$

APPENDIX B

**Proposition 1 (repeated).** Let $P$ and $G$ be a program and a closed goal of the modal
language $L_2$. If, for a given S4 interpretation $M = \langle W, R, \mathcal{D}, e \rangle$ and a world
$w \in W$ with every constant of $P$ in $\mathcal{D}(w)$, $M, w \models_{s_4} P \wedge \neg G$, then there is an
S4 interpretation $M_H = \langle W, R, \mathcal{D}_H, e_H \rangle$ with constant domain $U_p$ such that
$M_H, w \models_{s_4} P \wedge \neg G$.

**Proof.** Let $\mathcal{D}_H(w) = U_p$ and $e_H(w) = \{ p(a_1, \ldots, a_n) : p(a_1, \ldots, a_n) \in e(w)$ and
$a_1, \ldots, a_n \in U_p \}$ for all $w \in W$. In order to prove Proposition 1, we will prove that,
for each clause $D$ and closed goal $G$ in the language of $P$ and for all $w' \in W$ such that
$wRw'$, the following two statements hold:

1. If $M, w' \models_{s_4} G$, then $M_H, w' \models_{s_4} \neg G$,
2. If $M, w' \models_{s_4} D$, then $M_H, w' \models_{s_4} D$.

From (1) and (2), since the accessibility relation $R$ is reflexive and, therefore,$wRw'$, we
can derive the thesis.

In the following proof, we will make use of the fact that, by hypothesis, every
constant of $P$ is in $\mathcal{D}(w)$, i.e., $U_p \subseteq \mathcal{D}(w)$. Moreover, since the domains are
increasing, $\mathcal{D}(w) \subseteq \mathcal{D}(w')$ for all $w'$ such that $wRw'$. We prove (1) and (2) by
simultaneous induction on the structure of $D$ and $G$.

Let us first prove (1). We consider all the possible cases for $G$.

- If $G = T$, since $M, w' \models_{s_4} T$, (1) holds trivially.
- If $G = \Box p(a_1, \ldots, a_n)$, then
  $M, w' \models_{s_4} \neg \Box p(a_1, \ldots, a_n)$
  iff for some world $w'' \in W$ such that $w' Rw''$, $M, w'' \models_{s_4} p(a_1, \ldots, a_n)$
  iff for some world $w'' \in W$ such that $w' Rw''$, $p(a_1, \ldots, a_n) \in e(w'')$
  iff for some world $w'' \in W$ such that $w' Rw''$, $p(a_1, \ldots, a_n) \in e_H(w'')$
  (since $G$ is in the language of $P$, $a_1, \ldots, a_n \in U_p$)
  iff for some world $w'' \in W$ such that $w' Rw''$, $M_H, w'' \models_{s_4} \neg p(a_1, \ldots, a_n)$

- If $G = G_1 \wedge G_2$, then
  $M, w' \models_{s_4} \neg (G_1 \wedge G_2)$
  iff $M, w' \not\models_{s_4} G_1 \wedge G_2$
  iff $M, w' \not\models_{s_4} G_1$ or $M, w' \not\models_{s_4} G_2$
  $\Rightarrow M_H, w' \not\models_{s_4} G_1$ or $M_H, w' \not\models_{s_4} G_2$ [since $G_1$ and $G_2$ are subformulas
  of $G$, by inductive hypothesis, (1) holds for them]
  iff $M_H, w' \not\models_{s_4} G_1 \wedge G_2$
  \[\therefore M, \ldots, w' \vdash \neg (G_1 \wedge G_2)\]
• If $G = \exists x G'$, then  
  $M, w' \models_{S_4} \exists x G'$  
  iff $M, w' \not\models_{S_4} \exists x G'$  
  iff for all $c \in \mathcal{D}(w')$, $M, w' \not\models_{S_4} G'[c/x]$  
  $\Rightarrow$ for all $c \in U_p$, $M, w' \not\models_{S_4} G'[c/x]$ [since $U_p \subseteq \mathcal{D}(w) \subseteq \mathcal{D}(w')$]  
  $\Rightarrow$ for all $c \in U_p$, $M, w' \not\models_{S_4} G'[c/x]$ (by inductive hypothesis, since $G'[c/x]$ is in the language of $P$ and $wRw'$, (1) holds for $G'[c/x]$)  
  iff $M_H, w' \not\models_{S_4} \exists x G'$  
  iff $M_H, w' \models_{S_4} \neg \exists x G'$.

• If $G = \Box(D \supset G')$, then  
  $M, w' \models_{S_4} \Box(D \supset G')$  
  iff for some world $w'' \in W$: $w'Rw''$, $M, w'' \not\models_{S_4} D \supset G'$  
  iff for some world $w'' \in W$: $w'Rw''$, $M, w'' \models_{S_4} D$ and $M, w'' \not\models_{S_4} G'$  
  $\Rightarrow$ for some world $w'' \in W$: $w'Rw''$, $M_H, w'' \models_{S_4} D$ and $H_H, w'' \not\models_{S_4} G'$  
  (by inductive hypothesis, since both $D$ and $G'$ are subformulas of $G$)  
  iff for some world $w'' \in W$: $w'Rw''$, $M_H, w'' \not\models_{S_4} D \supset G'$  
  iff $M_H, w' \models_{S_4} \neg \Box(D \supset G')$.

Let us now prove (2). We consider all the possible cases for $D$.

• If $D = G \supset \Box p(a_1, \ldots, a_n)$, then  
  $M, w' \models_{S_4} G \supset \Box p(a_1, \ldots, a_n)$  
  iff $M, w' \not\models_{S_4} G$ or $M, w'' \models_{S_4} p(a_1, \ldots, a_n)$, for all worlds $w'' \in W$ such that $w'Rw''$  
  iff $M, w' \not\models_{S_4} G$ or $p(a_1, \ldots, a_n) \in e(w'')$, for all worlds $w'' \in W$ such that $w'Rw''$  
  $\Rightarrow M_H, w' \not\models_{S_4} G$ or $p(a_1, \ldots, a_n) \in e_H(w'')$, for all worlds $w'' \in W$ such that $w'Rw''$ [since $G$ is a subformula of $D$, (1) holds for $G$; moreover, since $D$ is in the language of $P$, $a_1, \ldots, a_n \in U_p$]  
  iff $M_H, w' \not\models_{S_4} G$ or $M_H, w'' \models_{S_4} p(a_1, \ldots, a_n)$, for all worlds $w'' \in W$ such that $w'Rw''$  
  iff $M_H, w' \models_{S_4} G \supset \Box p(a_1, \ldots, a_n)$.

• If $D = D_1 \land D_2$, then  
  $M, w' \models_{S_4} D_1 \land D_2$  
  iff $M, w' \models_{S_4} D_1$ and $M, w' \models_{S_4} D_2$  
  $\Rightarrow M_H, w' \models_{S_4} D_1$ and $M_H, w' \models_{S_4} D_2$ [since $D_1$ and $D_2$ are subformulas of $D$, by inductive hypothesis, (2) holds for them]  
  iff $M_H, w' \models_{S_4} D_1 \land D_2$.

• If $D = \forall x D'$, then  
  $M, w' \models_{S_4} \forall x D'$  
  iff for all $c \in \mathcal{D}(w')$, $M, w' \models_{S_4} D'[c/x]$  
  $\Rightarrow$ for all $c \in U_p$, $M, w' \models_{S_4} D'[c/x]$ [since $U_p \subseteq \mathcal{D}(w) \subseteq \mathcal{D}(w')$]  
  $\Rightarrow$ for all $c \in U_p$, $M_H, w' \models_{S_4} D'[c/x]$ (by inductive hypothesis, since $D'[c/x]$ is in the language of $P$, (2) holds for $D'[c/x]$)  
  iff $M_H, w' \models_{S_4} \forall x D'$.

Let us prove the analogue for the language $L_1$ with open dynamic blocks. In the following, let $L_1^\gamma$ be the language obtained by adding to $L_1$ countably many new constants. In the next theorem, we will consider Kripke interpretations in which $\mathcal{D}$ is a function from the set of worlds $W$ to a set of constants of $L_1^\gamma$.  


Proposition 2. Let $P$ and $G$ be a program and a closed goal of the modal language $L$. If, for a given S4 interpretation $M = \langle W, R, \mathcal{D}, e \rangle$ and a world $w \in W$ with every constant of $P$ in $\mathcal{D}(w)$, $M, w \models_{S4} P \land \neg G$, then there is an S4 interpretation $M_H = \langle W, R, \mathcal{D}_H, e_H \rangle$ with constant domain $U_p$ such that $M_H, w \models_{S4} P \land \neg G$.

Proof. Let $\mathcal{D}_H(w) = U_p$ and $e_H(w) = \{ p(a_1, \ldots, a_n) : p(a_1, \ldots, a_n) \in e(w) \text{ and } a_1, \ldots, a_n \in U_p \}$, for all $w \in W$. The proof is similar to the proof of Proposition 1. In language $L_1$, however, the definition of clauses is different w.r.t. language $L_2$: there is an additional $\Box$ in front of each clause definition and each universal quantifier. So we have to consider the cases $D = \Box \Box p(a_1, \ldots, a_n)$ and $D = \Box \forall x D'$, instead of the corresponding cases in the proof of Proposition 1.

- If $D = \Box \Box p(a_1, \ldots, a_n)$, then
  
  $M, w' \models_{S4} \Box \Box p(a_1, \ldots, a_n)$
  
  iff $M, w \models_{S4} \Box \Box p(a_1, \ldots, a_n)$, for all worlds $w \in W$ such that $w'Rw$
  
  iff $M, w \models_{S4} \Box \Box p(a_1, \ldots, a_n)$, for all worlds $w', w'' \in W$ such that $w'Rw'$ and $w''Rw''$
  
  iff $M, w \models_{S4} \Box \Box p(a_1, \ldots, a_n)$, for all worlds $w \in W$ such that $w'Rw'$ and $w''Rw''$
  
  iff $M_H, w \models_{S4} \Box \Box p(a_1, \ldots, a_n)$, for all worlds $w \in W$ such that $w'Rw'$ and $w''Rw''$
  
  iff $M_H, w \models_{S4} \Box \Box p(a_1, \ldots, a_n)$, for all worlds $w \in W$ such that $w'Rw'$ and $w''Rw''$.

- If $D = \Box \forall x D'$, then
  
  $M, w' \models_{S4} \Box \forall x D'$
  
  iff $M, w \models_{S4} \Box \forall x D'$, for all worlds $w \in W$ such that $w'Rw$
  
  iff for all $c \in \mathcal{D}(w)$, $M, w \models_{S4} D'[c/x]$, for all worlds $w \in W$ such that $w'Rw$
  
  \Rightarrow for all $c \in U_p$, $M, w \models_{S4} D'[c/x]$ for all worlds $w \in W$ such that $w'Rw$

  $[\text{since } U_p \subseteq \mathcal{D}(w) \subseteq \mathcal{D}(w')]$

  \Rightarrow for all $c \in U_p$, $M_H, w \models_{S4} D'[c/x]$ for all worlds $w \in W$ such that $w'Rw$

  $[\text{by inductive hypothesis, since } D'[c/x] \text{ is in the language of } P \text{ and } wRw \text{ by transitivity of } R]$

  $M_H, w \models_{S4} \Box \forall x D'$, for all worlds $w \in W$ such that $w'Rw$

  $M_H, w \models_{S4} \Box \forall x D'$. □

When Kripke interpretations with constant domain $U_p$ are considered for the language $L_1$, the modal operator $\Box$ in front of universal quantifiers can be eliminated. In this case, by replacing $\Box \forall x D$ with $\forall x D$ in the first-order modal language $L_1$, the language $\mathcal{L}_1$ is obtained:

\[
\mathcal{L}_1: \quad G := T \mid \Box A \mid G_1 \land G_2 \mid \exists x G \mid \Box (D \supset G),
\]

\[
D := \Box (G \supset \Box A) \mid D_1 \land D_2 \mid \forall x D.
\]
To prove that the language $\mathcal{L}_1$ is equivalent to the language $L_1$, we define a mapping $\circ$ from formulas in $L_1$ to formulas in $\mathcal{L}_1$, as follows:

\begin{align*}
T^\circ &= T, \\
(\Box A)^\circ &= \Box A, \\
(\exists x G)^\circ &= \exists x G^\circ, \\
(\Box (D \supset G))^\circ &= \Box (D^\circ \supset G^\circ), \\
(D_1 \land D_2)^\circ &= D_1^\circ \land D_2^\circ, \\
(\Box (G \supset \Box A))^\circ &= \Box (G^\circ \supset \Box A), \\
(\Box \forall x D)^\circ &= \forall x D^\circ.
\end{align*}

Let us first prove the following lemma.

**Lemma 3.** Let $P$ be a program in the modal language $\mathcal{L}_1$. Let $H_P = \langle W, R, \mathcal{D}, e \rangle$ be an S4 interpretation with constant domain $U_P$ [i.e., $\mathcal{D}(w) = U_P$, for all $w \in W$.

For all worlds $w \in W$ and for all programs $D$ in the language $\mathcal{L}_1$, with symbols in $P$,

$$M_H, w \models_{S4} D \iff M_H, w \models_{S4} \Box D.$$  

**Proof.** In the direction ($\Rightarrow$), it is obvious since $R$ is reflexive. We prove ($\Leftarrow$) by induction on the structure of $D$.

- If $D = \Box (G \supset \Box A)$, then

  $$M_H, w \models_{S4} \Box (G \supset \Box A) \implies M_H, w' \models_{S4} G \supset \Box A, \text{ for all worlds } w' \in W \text{ such that } wRw' \implies M_H, w'' \models_{S4} G \supset \Box A, \text{ for all worlds } w' \in W \text{ such that } wRw' \text{ and } w'' \sim Rw' \text{ (since } R \text{ is transitive)} \implies M_H, w \models_{S4} \Box (G \supset \Box A), \text{ for all worlds } w \in W \text{ such that } wRw.'$$

  $$M_H, w \models_{S4} \Box (G \supset \Box A) \text{ (i.e., } M_H, w \models_{S4} \Box D).$$

- If $D = \forall x D'$, then

  $$M_H, w \models_{S4} \forall x D' \implies \text{for all } t \in U_P, M_H, w \models_{S4} D'[t/x] \implies \text{for all } t \in U_P, M_H, w \models_{S4} D'[t/x] \text{ (by inductive hypothesis)} \implies \text{for all } t \in U_P, \text{ for all worlds } w' \in W \text{ such that } wRw', M_H, w' \models_{S4} D'[t/x] \implies \text{for all worlds } w' \in W \text{ such that } wRw', \text{ for all } t \in U_P, M_H, w' \models_{S4} D'[t/x] \implies \text{for all worlds } w' \in W \text{ such that } wRw', M_H, w' \models_{S4} \forall x D' \implies M_H, w \models_{S4} \forall x D' \text{ (i.e., } M_H, w \models_{S4} \Box D).$$

- If $D = D_1 \land D_2$, then

  $$M_H, w \models_{S4} D_1 \land D_2 \implies M_H, w \models_{S4} D_1 \text{ and } M_H, w \models_{S4} D_2 \implies M_H, w \models_{S4} \Box D_1 \text{ and } M_H, w \models_{S4} \Box D_2 \text{ (by inductive hypothesis)} \implies \text{for all worlds } w' \in W \text{ such that } wRw', M_H, w' \models_{S4} D_1 \text{ and for all worlds } w' \in W \text{ such that } wRw', M_H, w' \models_{S4} D_2 \implies \text{for all worlds } w' \in W \text{ such that } wRw', M_H, w' \models_{S4} D_1 \land D_2 \implies M_H, w \models_{S4} \Box (D_1 \land D_2) \text{ (i.e., } M_H, w \models_{S4} \Box D).$$
In particular, by Lemma 3, we have that, for all worlds \( w \in W \) and for all formulas \( \forall x D \) in the language \( \mathcal{L}_1 \), \( M_{H_w}, w \models_{S_4} \forall x D \) iff \( M_{H_w}, w \models_{S_4} \Box \forall x D \). We will make use of this equivalence in the proof of the next proposition.

**Proposition 3.** Let \( P \) be a program in the modal language \( L_1 \). Let \( M_{H_w} = \langle W, R, \mathcal{P}, e \rangle \) be an S4 interpretation with fixed domain \( U_p \) [i.e., \( \mathcal{P}(w) = U_p \), for all \( w \in W \)]. For all worlds \( w \in W \) and for all programs \( D \) and closed goals \( G \) in the language \( L_1 \), with symbols in \( P \):

1. \( M_{H_w}, w \models_{S_4} G \Leftrightarrow M_{H_w}, w \models_{S_4} G^o \),
2. \( M_{H_w}, w \models_{S_4} D \Leftrightarrow M_{H_w}, w \models_{S_4} D^o \).

**Proof.** (i) and (ii) can be proved by simultaneous induction on the structure of \( D \) and \( G \).

Let us first prove (i):

- If \( G = T \) and \( G = \Box A \), (i) holds obviously, since \( T^o = T \) and \( (\Box A)^o = \Box A \).
- If \( G = G_1 \land G_2 \), then
  \[ M_{H_w}, w \models_{S_4} G_1 \land G_2 \]
  iff \( M_{H_w}, w \models_{S_4} G_1 \) and \( M_{H_w}, w \models_{S_4} G_2 \)
  iff \( M_{H_w}, w \models_{S_4} G_1^o \) and \( M_{H_w}, w \models_{S_4} G_2^o \) [since \( G_1 \) and \( G_2 \) are subformulas of \( G \), by inductive hypothesis, (i) holds for them]
  iff \( M_{H_w}, w \models_{S_4} G_1^o \land G_2^o \) (by definition of the mapping \( ^o \)).

- If \( G = \exists x G_1 \), then
  \[ M_{H_w}, w \models_{S_4} \exists x G_1 \]
  iff for some \( t \in U_p \), \( M_{H_w}, w \models_{S_4} G_1[t/x] \)
  iff for some \( t \in U_p \), \( M_{H_w}, w \models_{S_4} (G_1[t/x])^o \) (by inductive hypothesis)
  iff \( M_{H_w}, w \models_{S_4} \exists x G_1^o \) (by definition of the mapping \( ^o \)).

- If \( G = \Box (D \supset G_1) \), then
  \[ M_{H_w}, w \models_{S_4} \Box (D \supset G_1) \]
  iff for all worlds \( w' \in W \): \( wRw', M_{H_w}, w' \models_{S_4} D \supset G_1 \)
  iff for all worlds \( w' \in W \): \( wRw', M_{H_w}, w' \not\models_{S_4} D \) or \( M_{H_w}, w' \models_{S_4} G_1 \)
  iff for all worlds \( w' \in W \): \( wRw', M_{H_w}, w' \not\models_{S_4} D^o \) or \( M_{H_w}, w' \models_{S_4} G_1^o \) (by inductive hypothesis, since both \( D \) and \( G_1 \) are subformulas of \( G \))
  iff for all worlds \( w' \in W \): \( wRw', M_{H_w}, w' \models_{S_4} G_1^o \)
  iff \( M_{H_w}, w \models_{S_4} \Box (D^o \supset G_1^o) \)
  iff \( M_{H_w}, w \models_{S_4} (\Box (D \supset G_1))^o \) (by definition of the mapping \( ^o \)).

Let us now prove (ii):

- If \( D = \Box (G \supset \Box A) \), then
  \[ M_{H_w}, w \models_{S_4} \Box (G \supset \Box A) \]
  iff \( M_{H_w}, w' \models_{S_4} G \supset \Box A \), for all worlds \( w' \in W \) such that \( wRw' \)
  iff \( M_{H_w}, w' \not\models_{S_4} G \) or \( M_{H_w}, w' \models_{S_4} \Box A \), for all worlds \( w' \in W \) such that \( wRw' \)
  (by inductive hypothesis, since \( G \) is a subformula of \( D \)).
iff \( M_{H}, w' \models_{\mathcal{S}_{4}} G \supset \Box A \), for all worlds \( w' \in W \) such that \( wRw' \)

- If \( D = D_{1} \land D_{2} \), then we proceed as for \( G = G_{1} \land G_{2} \).

- If \( D = \Box \forall xD_{1} \), then

\[
M_{H}, w \models_{\mathcal{S}_{4}} \Box \forall xD_{1}
\]

iff \( M_{H}, w' \models_{\mathcal{S}_{4}} \forall xD_{1} \), for all worlds \( w' \in W \) such that \( wRw' \)

iff for all \( t \in Up, M_{H}, w' \models_{\mathcal{S}_{4}} D_{1}[t/x] \), for all worlds \( w' \in W \) such that \( wRw' \)

iff for all \( t \in Up, M_{H}, w' \models_{\mathcal{S}_{4}} (D_{1}[t/x])^{o} \), for all worlds \( w' \in W \) such that \( wRw' \)

(by inductive hypothesis)

iff \( M_{H}, w' \models_{\mathcal{S}_{4}} \forall xD_{1}^{o} \) (by Lemma 3, since \( \forall xD_{1}^{o} \) is a formula in \( \mathcal{L}_{1} \))

iff \( M_{H}, w \models_{\mathcal{S}_{4}} (\Box \forall xD_{1})^{o} \) (by definition of the mapping \( o \)).

We are grateful to Dale Miller for the stimulating discussions on the subject of this paper. Also, we wish to thank the anonymous referees for their careful checking of the work and their helpful comments.

REFERENCES


