On Reflection and Meta-Level Architecture and their Applications in AI

Mamdouh Ibrahim, Chair

WORKING NOTES

MONDAY, AUGUST 21, 1995
Montréal, Québec
On Reflection and Meta-Level Architecture and their Applications in AI

Mamdouh Ibrahim, Chair

Working Notes

Monday, August 21, 1995
Montréal, Québec
A DYNAMIC LOGIC FRAMEWORK FOR REFLECTIVE ARCHITECTURES

Carles Sierra, Lluís Godo, Ramon López de Mántaras

Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
Campus de la UAB, 08193 Cerdanyola, Catalonia, Spain
sierra@iiia.csic.es, http://www.iiia.csic.es

ABSTRACT

The aim of this paper is to use Dynamic Logic as a common logical framework to describe and identify the most relevant formal characteristics of Reflective Logical Architectures (RLA) in order to investigate the expressive power of the KBSs that can be built upon them, from now on called Reflective Knowledge Bases (RKB). In general, a RLA allows to build RKBs as a set of units with initial local theories written in possibly different logical languages. Each unit is also usually allowed to have its own intra-unit deductive system. Moreover, the whole RKB is equipped with an additional set of deductive rules, called Reflection rules, to control the information flow among the different units of the RKB. The set of Reflection rules acts as an inter-unit deductive system. The reasoning dynamics of a RKB on top of a RLA can therefore be described by how the local theories of the units evolve during execution.

1 INTRODUCTION

In Artificial Intelligence, Reflective Logical Architectures, RLA for short, CMC [5,6,7], RWS [14], MIRA [12], DESIRE [12], MILORD II [1,2,10], Omega [9], FOL[15] are particular types of architectures to build knowledge-based systems (KBS) that play a major role in dealing with complex reasoning patterns, such as those involved in non-monotonic reasoning, scheduling or planning. Despite the fact that many commonalities can be intuitively found[14], there is a lack of a formal framework to compare and describe them. These architectures are based on the use of several logical languages to define local theories (or meta-theories acting upon theories) that influence/modify each other. These influences are modelled by complex control patterns of the reasoning flow between system units (also called modules or contexts) containing different knowledge theories evolving in time. Furthermore, the control patterns are often dynamically changed at run time. Therefore, when trying to define a formal framework to describe reflective architectures it is mandatory that such a framework be able to model this dynamic behaviour.

Dynamic logic [8] has been traditionally used to describe and compare dynamic systems. Particularly, it has been used to deal with computational systems, understanding computations of programs as dynamic state changes. The aim of this kind of logic is the study of the mathematical properties of programs, and their behaviour. The construction of tools to reason about programs, and the discovery of the key concepts involved in this reasoning process are the long term research in the field.

In this paper we propose a particular propositional dynamic logic called Reflective Dynamic Logic (RDL from now on) in such a way that a RKB, built within a concrete RLA, will be transformed into a theory in RDL. The atomic RDL-formulas will be just the formulas of the set of languages of the RLA upon which the RKB is built. The key point is what an state and a program are mean in the RDL framework. In the dynamic logic terminology, a state is a complete description of which atoms are true and which atoms are false in RDL. A change in unit local theories will occur by the action of its local deductive system. Axiomatics of RLAs, written in RDL, will restrict which local theory transitions are allowed; therefore, in this setting atomic programs will be the elementary deductive steps of any deduction system (intra or inter-unit), whereas the semantics of compound programs will represent any possible control reasoning flow combining as many atomic programs as necessary.

The following interpretation allows to map proofs in RKBs into formulas in RDL:

94
On Reflection and Meta-Level Architecture and their Applications in AI

Mamdouh Ibrahim, Chair

Working Notes

Monday, August 21, 1995
Montréal, Québec
A DYNAMIC LOGIC FRAMEWORK FOR REFLECTIVE ARCHITECTURES

Carles Sierra, Lluís Godo, Ramon López de Mántaras

Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
Campus de la UAB, 08193 Cerdanyola, Catalonia, Spain
sierra@iiia.csic.es, http://www.iiia.csic.es

ABSTRACT

The aim of this paper is to use Dynamic Logic as a common logical framework to describe and identify the most relevant formal characteristics of Reflective Logical Architectures (RLA) in order to investigate the expressive power of the RKBs that can be built upon them, from now on called Reflective Knowledge Bases (RKB). In general, a RLA allows to build RKBs as a set of units with initial local theories written in possibly different logical languages. Each unit is also usually allowed to have its own intra-unit deductive system. Moreover, the whole RKB is equipped with an additional set of deductive rules, called Reflection rules, to control the information flow among the different units of the RKB. The set of Reflection rules act as an inter-unit deductive system. The reasoning dynamics of a RKB on top of a RLA can therefore be described by how the local theories of the units evolve during execution.

1 INTRODUCTION

In Artificial Intelligence, Reflective Logical Architectures, RLA for short, (MC [5,6,7], RMS [11], ML [4]) DESIRE [12], MILORD II [1,2,16], OMEGA[9], FOL[15]) are particular types of architectures to build knowledge-based systems (KBS) that play a major role in dealing with complex reasoning patterns, such as those involved in non-monotonic reasoning, scheduling or planning. Despite the fact that many commonalities can be intuitively found[14], there is a lack of a formal framework to compare and describe them. These architectures are based on the use of several logical languages to define local theories (or meta-theories acting upon theories) that influence/modify each other. These influences are modeled by complex control patterns of the reasoning flow between system units (also called modules or contexts) containing different knowledge theories evolving in time. Furthermore, the control patterns are often dynamically changed at run time. Therefore, when trying to define a formal framework to describe reflective architectures it is mandatory that such a framework be able to model this dynamic behaviour.

Dynamic logic [8] has been traditionally used to describe and compare dynamic systems. Particularly, it has been used to deal with computational systems, understanding computations of programs as dynamic state changes. The aim of this kind of logic is the study of the mathematical properties of programs, and their behaviour. The construction of tools to reason about programs, and the discovery of the key concepts involved in this reasoning process are the long term research in the field.

In this paper we propose a particular propositional dynamic logic called Reflective Dynamic Logic (RDL from now on) in such a way that a RKB, built within a concrete RLA, will be transformed into a theory in RDL. The atomic RDL-formulas will be just the formulas of the set of languages of the RLA upon which the RKB is built. The key point is what an state and a program are meant to be in this logical logic. In the dynamic logic terminology, a state is a complete description of which atoms are true and which atoms are false in RDL. A change in unit local theories will occur by the action of its local deductive system. Axiomatics of RLAs, written in RDL, will restrict which local theory transitions are allowed; therefore, in this setting atomic programs will be the elementary deductive steps of any deduction system (intra or inter-unit), whereas the semantics of compound programs will represent any possible control reasoning flow combining as many atomic programs as necessary.

The following interpretation allows to map proofs in RKBs into formulas in RDL:

94
i) elementary computational steps of RKBs as atomic programs in RDL,

ii) local theories of the RKBs as formulas of RDL, and

iii) operational semantics of RKBs as axioms in RDL.

This means that execution of a RKB is equivalent to deduction in RDL. In this context RDL can be understood as a formal basis for comparison between several approaches to meta-level reasoning, as well as a specification language for RKBs. As an illustration of this approach a small example is described using RDL. Moreover some initial ideas about the description of well-known architectures will be given.

The paper is structured as follows. Section 2 contains a brief remainder of Propositional Dynamic Logic. Section 3 presents a formalization of Reflective Logical Architectures and related notions. In sections 4-6 a particular dynamic logic called RDL, used to model RLAs is proposed. Finally, in Section 7 the mapping of Reflective Knowledge-bases into RDL is exemplified.

2 SYNTAX AND SEMANTICS FOR PDL

In this section we make a short review of propositional dynamic logic (PDL), for a detailed description see [8].

General Syntax for PDL

Given a set of propositional atomic variables \( \Phi_0 \) and atomic programs \( \Pi_0 \), the set \( \Phi \) of compound formulas and the set \( \Pi \) of compound programs of RDL are defined as:

(i) \( \text{true} \in \Phi; \text{false} \in \Phi; \Phi_0 \subseteq \Phi \);

(ii) if \( p, q \in \Phi \) then \( \neg p \in \Phi \) and \( (p \lor q) \in \Phi \),

(iii) if \( p \in \Phi \) and \( \alpha \in \Pi \) then \( \langle \alpha \rangle p \in \Phi \),

(iv) \( \Pi_0 \subseteq \Pi \);

(v) if \( \alpha \in \Pi \) and \( \beta \in \Pi \) then \( (\alpha \land \beta) \in \Pi \), \( (\alpha \lor \beta) \in \Pi \)

and \( \alpha^e \in \Pi \).

(vi) if \( p \in \Phi \) then \( p^e \in \Pi \)

\( [\alpha]p \) is the usual modal abbreviation for \( \neg \langle \alpha \rangle \neg p \). Also, \( \land, \lor, \rightarrow, \leftrightarrow \) are abbreviations with the standard meaning. The meaning of the compound programs is the usual one [8].

General Semantics for PDL

The semantics of PDL is defined relative to a structure \( \mathcal{M} \) of the form \( \mathcal{M} = (W, \tau, p) \) where \( W \) is a set of states, \( \tau \) a mapping assigning to each formula \( \phi \) the set of states in which \( \phi \) is true, that is, \( \tau: \Phi \rightarrow 2^W \) and \( p \) a mapping which assigns to each program a set of pairs (s, t) representing transitions between states, that is, \( p: \Pi \rightarrow 2^{W \times W} \). More concretely, the mappings \( \tau \) and \( p \) are defined in [8] as follows:

\[
\begin{align*}
\tau(\text{true}) &= W \\
\tau(\text{false}) &= \emptyset \\
\tau(\neg \phi) &= W - \tau(\phi) \\
\tau(\phi \lor \psi) &= \tau(\phi) \cup \tau(\psi) \\
\tau(\langle \alpha \rangle \phi) &= \{s \in W \mid \exists t \ ((s,t) \in \tau(\phi) \land t \in \tau(\phi)) \} \\
p(\alpha \beta) &= \{ (s,t) \mid \exists u \ ((s,u) \in p(\alpha) \land (u,t) \in p(\beta)) \} \\
p(\alpha \lor \beta) &= p(\alpha) \lor p(\beta) \\
p(\alpha^e) &= \{ (s,t) \mid \exists k, s_0, \ldots, s_k (s_0 = s \land s_k = t \land \\
&\forall 1 \leq i \leq k (s_i, s_i) \in p(\alpha)) \} \\
p(\neg \phi) &= \{ (s,s) \mid s \in \tau(\phi) \}
\end{align*}
\]

As for notions of satisfiability and validity we write:

* \( (\mathcal{M}, s) \models p \), and we say that \( p \) is true in \( s \) or that \( s \)

satisfies \( p \), if \( s \in \tau(\phi) \).

* \( \mathcal{M} \models p \), and we say that \( p \) is \( \mathcal{M} \)-valid, iff \( (\mathcal{M}, s) \models p \) \( \forall s \in W \).

* \( \models p \), and we say that \( p \) is valid iff \( p \) is \( \mathcal{M} \)-valid for every

structure \( \mathcal{M} \).

A proposition \( \phi \) is said to be a global logical consequence of a set of formulas \( \Gamma \) if for any structure \( \mathcal{M} \), we have that \( \mathcal{M} \models \phi \) whenever \( \mathcal{M} \models \Gamma \), in which case we write \( \Gamma \models_{G} \phi \). A proposition \( \phi \) is said to be a local logical consequence of a set of formulas \( \Gamma \) if for any structure \( \mathcal{M} \) and state \( s \), we have that \( (\mathcal{M}, s) \models \phi \) whenever \( (\mathcal{M}, s) \models \Gamma \) in which case we write \( \Gamma \models_{L} \phi \).

Axiomaties for PDL

The next set of axioms define the propositional dynamic logic [8]:

\[
\begin{align*}
(A1) \text{ All instances of tautologies of the propositional calculus} \\
(A2) \langle \alpha \rangle (p \lor q) &\leftrightarrow (\langle \alpha \rangle p \lor \langle \alpha \rangle q) \\
(A3) \langle \alpha \rangle \langle \beta \rangle p &\leftrightarrow (\langle \alpha \rangle \beta) p \\
(A4) \langle \alpha \cup \beta \rangle p &\leftrightarrow (\langle \alpha \rangle \langle \beta \rangle p) \\
(A5) \langle \alpha^e \rangle p &\leftrightarrow (\langle p \rangle \langle \alpha^e \rangle p) \\
(A6) \langle \neg \rangle \neg q &\leftrightarrow p \lor q \\
(A7) [\alpha^e] \langle \langle \alpha \rangle \rightarrow q \rangle &\rightarrow ([\langle \alpha \rangle]q) \\
(A8) [\alpha] (p \rightarrow q) &\rightarrow ([\alpha]p \rightarrow [\alpha]q)
\end{align*}
\]

The two only inference rules of PDL are:

\[
\begin{align*}
(MP) p, p \rightarrow q \rightarrow q &\quad (\text{Modus Ponens}) \\
(G) p \rightarrow [\alpha]p &\quad (\text{Generalization})
\end{align*}
\]

It is well-known that axioms (A1) through (A8), together with the modus ponens and the generalization
inference rules, are a complete axiomatization of PDL tautologies with respect to the above semantics [8].

3 REFLECTIVE LOGICAL ARCHITECTURES

The basic notion that we want to express by a "reflective logical architecture" is that of a logical system that provides as basic tools: 1) a set of possibly different logical languages, 2) a set of inference rules within each language, 3) a set of inference rules between some of these languages, 4) the possibility of defining a set of units, i.e. theories written in a particular language together with a particular set of inference rules, and 5) a set of possible interconnections of units through reflection rules, respecting some topological criteria, e.g. a binary tree structure, an acyclic graph, etc. This generalization is related to the concept of Multi-Context systems[5], MC for short, because MC systems are the most general approach to reflective architectures. We shall use whenever is possible the naming of MC systems: Unites correspond to Contexts in MC systems and Reflection rules correspond to Bridge Rules in MC systems.

In the next definitions we introduce the three concepts involved in the definition of a reflective knowledge-based system. Namely, what we call Reflective Logical Architecture (in the highest level of abstraction) represents the most general characteristics of our target computational systems. Then, a Reflective Knowledge-Based Structure specifies a subclass of those systems by fixing its components, that is the set of units, their languages, the inference rules among them and the topology. Finally, a Reflective Knowledge-Based System is the specification of a concrete system obtained by filling the units of a Reflective Knowledge-Based Structure with particular domain theories.

Definition 3.1 A Reflective Logical Architecture is a 4-tuple $\text{RLA} = (L, \Delta, S, T)$, where:

1) $L = \{L_j | j \in \mathbb{K}\}$ is a set of logical languages.
2) $\Delta = \{\Delta_{ij} | i, j \in \mathbb{K}\}$ is a set of inference rules between pairs of languages, i.e. $\Delta_{ij} \subseteq L_i \times L_j$. In particular, when $i = j$, $\Delta_{ii}$ denotes a set of inference rules relating formulas of the corresponding language; otherwise it denotes a set of bridge rules between two different languages.
3) $S$ is a finite set of symbols for unit identifiers.
4) $T$ is the set of possible topologies. Each topology is determined by a set of directed links between symbols from $S$, i.e. $T$ is a subset of $2^{\mathbb{S} \times \mathbb{S}}$.

It is possible to further generalize this definition by allowing inference rules with premises in different languages, that is having $\Delta = \{\Delta_{ij} | i, j \in \mathbb{K}\}$, where

$$\bigcup_{h, j \in \mathbb{K}} L_h \subseteq 2^{\bigcup_{i=1}^n J_i} \times L_h$$

Definition 3.2 A Reflective Knowledge-Based Structure RKB-ST for a given RLA is a 5-tuple $\text{RKB-ST} = (\text{RLA}, U, M_L, M_A, B)$ where:

1) $\text{RLA} = (L, \Delta, S, T)$ is a Reflective Logical Architecture.
2) $U = \{u_i | i \in \mathbb{K}\}$ is a set of unit identifiers, i.e. $U$ is a subset of $S$.
3) $M_L$ assigns a language to each unit identifier, i.e. $M_L: U \rightarrow L$.
4) $M_A$ assigns a set of inference rules to each unit identifier, i.e. $M_A: U \rightarrow \{\Delta_{ij} | i, j \in \mathbb{K}\}$ such that if $M_L(u) = L_j$ for some $j \in J$, then $M_A(u) \subseteq \Delta_{ij}$.
5) $B$ is a mapping that assigns a set of directed bridge rules to pairs of different units, i.e. $B: U \times U \rightarrow \Delta$ such that:
   
   (i) if $u_1 \neq u_2$, $U_L(u_1) = L_i$ and $U_L(u_2) = L_j$ then $B(u_1, u_2) \subseteq \Delta_{ij}$.
   
   (ii) $B(u, u) = \emptyset$, for any $u \in U$.
   
   (iii) $\{B(u_1, u_2) | B(u_1, u_2) \neq \emptyset \} \in T$, that is, the topology of RKB-ST is in accordance with the allowed topologies in RLA.

Notice that in this definition, even in the case where $u_1 \neq u_2$, $B(u_1, u_2)$ can be empty, denoting that unit $u_1$ has no (directed) link with the unit $u_2$. In this way, a unit $u_1$ is connected to a unit $u_2$ whenever $B(u_1, u_2) \neq \emptyset$.

Definition 3.3 A Reflective Knowledge-Based System RKB for a given RKB-ST is a 3-tuple $\text{RKB} = (\text{RKB-ST}, M_\Sigma, M_\Omega)$ where:

1) $\text{RKB-ST} = (\text{RLA}, U, M_L, M_A, B)$ is a Reflective Knowledge-Based Structure, and
2) $M_\Sigma$ assigns a concrete signature $M_\Sigma(u) = \{\mathsf{Const} \Rightarrow \mathsf{Sort}\}$ for the language $M_L(u)$ of each unit identifier $u$, such that $\mathsf{Func}; \mathsf{Oper} \Rightarrow \mathsf{Sort}$, gives a type, Sort, for each element in the alphabet Oper.
3) $M_\Omega$ assigns a set of formulas (initial local theory) built upon $M_\Sigma$ to each unit identifier, i.e. $M_\Omega: U \rightarrow \bigcup_{j \in \mathbb{K}} \Delta_{ij}$ such that if $M_L(u) = L_j$ then $M_\Omega(u) \subseteq \Delta_{ij}$.

As a consequence of the execution of a RKB, the local theory associated to $u_i$ will change due to the intra and inter-unit deductions. A reflection rule $r \in B(u_i, u_j)$ is a bridge rule between unit $u_i$ and unit $u_j$ whose premises are in language $L_i = M_L(u_i)$ and the consequent is in a (possibly different) language $L_j = M_L(u_j)$, that is $r \subseteq L_i \times L_j$. Notice that the sets of reflection rules can be empty. An empty set $B(u_i, u_j) =$
\( \emptyset \) means that no communication between units \( u_i \) and 
\( u_j \) is defined. An empty set \( M_A(u_i) = \emptyset \) means that the
local theory attached to unit \( u_i \) will evolve only if inter-
unit reflection rules leading to the unit are defined, 
otherwise the unit will remain static along the 
execution process of the RKB.

The distinction between architecture and reflective
knowledge base structure is in some cases not as sharp
as implied by the above definitions. Often some
inference rules are already fixed in the architecture
(e.g. DESIRE, MILORE II), and the number and/or
structure and/or role of units that can be built is also
fixed (e.g. DESIRE, MILORE II, ML\(^2\)). So, these
definitions have to be considered as a very general
approach to RLA and RKB descriptions.

The next example illustrates the way this notation
can be used to formalize the BMS architecture [11] as a
Reflective Knowledge-Based Structure named RKB-
ST\(_{BMS} \).

**Example 3.4.** Let be \( RLA^1 = (I, \Delta, S, T) \), where each
component is described as follows.

- \( I=\{PL, FOL\} \), where \( PL \) is a propositional language
  with connectives: \( \neg, \&,-,\subset \) and sentences are pairs of
  classical propositional formulas and truth values in the
  set \( \{0,1\} \), meaning true, false and unknown
  respectively. \( FOL \) is a first order language with
  predicate signature \( (T, PA) \) where \( T(s) \) means that \( s \)
  is true and \( PA(s) \) means that \( s \) is a possible assumption.
  Constants of \( FOL \) are propositional variables of \( PL \).

- \( \Delta=\{\Delta_{PL}, \Delta_{FOL}, \Delta_{FOL}, \Delta_{FOL}, \Delta_{FOL}, \Delta_{PL}, \Delta_{FOL} \} \). \( \Delta_{PL}, \Delta_{FOL} \)
  and \( \Delta_{FOL} \) contain the instances of the *modus
  ponens* and *introduction*, and possibly others inference
  rules. Bridge rules are:

\[
\begin{align*}
\Delta_{PL,PL} &= \left\{ \left( \phi, 1 \right), \left( \phi, u \right), \left( \phi, 0 \right) \right\}, \\
\Delta_{PL,PL} &= \left\{ \left( T(\varphi) \right), \neg T(\varphi), \neg T(\varphi) \right\} \\
\Delta_{FOL,PL} &= \left\{ \left( PA(\varphi) \right), (\varphi, 1) \right\} \\
\end{align*}
\]

- \( S \) is the usual set of symbol identifiers and \( T \) the
  possible topologies, consisting of all possible pairs of
  unit interconnections.

The corresponding RKB-ST of the BMS system, which
is a particular structure over the above architecture,
contains just a couple of units. Therefore,

\[
RKB-ST_{BMS} = (RLA, U, M_L, M_A, B),
\]

- \( U = \{a, b\} \),
- \( M_L = \{ a \rightarrow PL, b \rightarrow FOL \} \),
- \( M_A = \{ (a, a) \rightarrow PL, (b, b) \rightarrow FOL \} \), and
- \( B = \{ (a, b) \rightarrow PL, (b, a) \rightarrow FOL \} \).

4 QUOTING RKB-STS IN PDL: A REFLECTIVE
DYNAMIC LOGIC

Following the approach of previous section, we shall
present a methodology to define a particular PDL
capturing the basic structure of a RKB-ST, that is units,
languages, inference rules and topology and leaving out
the particular initial sets of formulas in each unit.
These local sets of formulas will be translated as a
theory of PDL formulas built on top of the logics we are
going to define next. Such logics will be generically
called RDL (Reflective Dynamic Logic). To build a
RDL we need first of all to fix the set of atomic
formulas and the set of programs. Given a RKB-ST =
(\( RL, U, M_L, M_A, B \), the set of atomic formulas of
RDL will be defined as the set of "quoted" formulas
from the languages \( L \) in \( RLA \), indexed by the unit
identifiers in \( U \), and the set of RDL atomic programs
will be restricted to represent deduction steps in those
languages. This is formalised in the following
definitions.

**Definition 4.1** Given a RKB-ST, the set of atomic
formulas of RDL is defined as

\[
\Phi_0 = \{ u \cdot [\varphi] \mid u \in U, \varphi \in M_L(u) \}
\]

that is, formulas are indexed by unit names, using the
notation unit identifier-quoted formula.

**Definition 4.2** Given a RKB-ST, the set \( \Pi_0 \) of atomic
programs of RDL is defined as the union of the intra-
unit inference rules \( \Pi_{0,\text{intra}} \) and the RKB-ST reflection
rules \( \Pi_{0,\text{inter}} \):

\[
\begin{align*}
\Pi_0 &= \Pi_{0,\text{intra}} \cup \Pi_{0,\text{inter}} \\
\Pi_{0,\text{intra}} &= \bigcup_{k \in K} \Pi_{0,\text{intra}}^{k} \\
\Pi_{0,\text{inter}} &= \bigcup_{k \in K} \Pi_{0,\text{inter}}^{k} \\
\Pi_{0,\text{intra}}^{k} &= \Gamma \cup \{ \varphi \mid \Gamma \cup \{ \varphi \} \subseteq M_L(u_k) \}, \\
\Pi_{0,\text{inter}}^{k} &= \Gamma \cup \{ \varphi \mid \varphi \in M_L(u_k) \}, \\
\end{align*}
\]

Notice that programs contain not only the inference
rules applied but also the formulas involved in the
deduction steps. We can now define the compound
programs modeling entailments intra or inter-units.
Next definition introduces indeterminism in the entailment via the union of programs representing inference rules.

**Definition 4.3** Given the above set of atomic programs, we define the next couple of sets of indeterministic compound programs standing for inter and intra-unit deductions:

- **Intra-unit deduction** is $\Gamma_{\alpha} = \bigcup_{\alpha \in \Pi} \alpha_{\text{intra}}$

- **Inter-unit deduction** is $\Gamma_{\alpha} = \bigcup_{\alpha \in \Pi} \alpha_{\text{inter}}$

The following useful notation will be used in the example of section 7:

$$\alpha^c = \alpha^c; (\bigwedge_{\varphi \in \Phi} (\varphi > \varphi \rightarrow \varphi)),$$

where $\alpha^c$ represents the closure of program $\alpha$. That is, this program will lead to an state in which no different state is reachable by another application of program $\alpha$. Next proposition establishes the correct behaviour of closure of programs. It will be used in the example of section 7.

**Proposition 4.4** The next formula is valid in PDL:

$$\varphi \rightarrow [\alpha^c] \varphi \quad \forall \alpha \in \Pi$$

5 SEMANTICS OF RDL: THE INTENDED MODELS

The semantics of RDL is defined, as in PDL, relative to a structure $\mathcal{M}$ of the form $\mathcal{M} = (W, \tau, \rho)$, as a particularization of the general semantics for PDL by adding a particular interpretation of both the RDL atomic formulas $\Phi_0$ and the RDL atomic programs $\Pi_0$.

The target computational systems we are trying to model in RDL is very wide, showing very different behaviours from a logical point of view. Some of these systems are non-monotonic, some are conservative, etc. Given this fact, the best way of giving semantics to them is via the definition of a hierarchy of classes of Kripke structures going from the more general or less restrictive ones to the more particular ones. In this sense we will define four particular interpretations of the $\rho$ mapping while keeping the $\tau$ interpretation in a standard way.

We will consider a particular Reflective Knowledge Base structure $\text{RKB-ST} = (\text{RLA}, U, M_L, M_A, B)$ fixed from now on.

The next class of models is the basic one, in the sense that, for a given deduction represented by an atomic program, it restricts the possible transitions between states to those whose premises are true in the source state and whose conclusion is made true by the program in the target state.

**Class of models $\mathcal{M}_1$** consists of structures $(W, \tau, \rho_1)$ where the mapping $\rho_1$ is defined as follows:

$$\rho_1([\Gamma\varphi]) = \{(s, t) | s, t \in W, \tau = \{\varphi\}, s \in \tau(k - [\varphi]), t \in \tau(l - [\varphi])\}, \text{ if } [\Gamma\varphi] \in \Pi_0$$

Further conditions to be imposed to possible transitions, that are usually fulfilled by the architectures under consideration in this paper, are the following:

- a formula different from the conclusion in the program and true in the target state must also be true in the source state.
- a formula is true in the target state whenever it is true in the source one.

These conditions lead to the two following classes of models.

**Class of models $\mathcal{M}_2$** consists of structures $(W, \tau, \rho_2)$ where the mapping $\rho_2$ is defined as follows:

$$\rho_2([\Gamma\varphi]) = \{(s, t) | s, t \in \rho_2([\Gamma\varphi]), \forall \varphi \in M_L(u_i) : \varphi \neq \varphi, t \in \tau(l - [\varphi]) \Rightarrow s \in \tau(l - [\varphi]), \text{ if } [\Gamma\varphi] \in \Pi_0$$

**Class of models $\mathcal{M}_3$** consists of structures $(W, \tau, \rho_3)$ where the mapping $\rho_3$ is defined as follows:

$$\rho_3([\Gamma\varphi]) = \{(s, t) | s, t \in \rho_3([\Gamma\varphi]), \forall \varphi \in M_L(u_i) : s \in \tau(l - [\varphi]) \Rightarrow t \in \tau(l - [\varphi]), \text{ if } [\Gamma\varphi] \in \Pi_0$$

Finally, taking both previous conditions together leads to the following fourth class of models in which the true formulas in the target state are just those that are true in the source state plus the formula being deduced by the program.
Class of models $\mathcal{M}_\mathcal{P}$. It consists of structures $(W, \tau, p_4)$ where the mapping $p_4$ is defined as follows:

\[ p_4(\Gamma \cup \varphi) = p_4(\Gamma \cup \varphi) \cap p_4(\Gamma \cup \varphi). \]

Obviously, these classes of models respect the next inclusions:

$\mathcal{M}_4 \subseteq \mathcal{M}_3 \subseteq \mathcal{M}_2$

$\mathcal{M}_4 \subseteq \mathcal{M}_3 \subseteq \mathcal{M}_2$

and therefore the validities with respect to these classes of models go in the reverse direction.

**Example 5.1** In the case where unit $k$ is endowed with the modus ponens inference rule, it is easy to show the $\mathcal{M}_\mathcal{P}$-validity of the RDL formula modelling such local inference rule, i.e.:

$\mathcal{M}_\mathcal{P} \vdash (\beta(\neg \varphi) \land k \rightarrow (\beta \rightarrow k \rightarrow \gamma)) \rightarrow (k \rightarrow k \rightarrow \gamma)$

where the modus ponens program $\vdash \neg \varphi$ is defined as:

$\vdash \neg \varphi = \bigcup \{ (\psi, \varphi \rightarrow \varphi) \mid \varphi \in \mathcal{L}_k \}$

**Example 5.2** Next formulas is $\mathcal{M}_\mathcal{P}$-valid in RDL. It represents the commutativity of non-related reflective deduction steps.

$\mathcal{M}_\mathcal{P} \vdash (\alpha \vdash^{j \rightarrow k} \beta) \psi \leftrightarrow (\alpha \vdash^{k \rightarrow j} \beta) \psi$, for $j \neq k, i \neq l$

**Proposition 5.3.** The next $\mathcal{M}_\mathcal{P}$-validities hold:

$\mathcal{M}_\mathcal{P} \vdash (i \rightarrow \varphi) \rightarrow (j \rightarrow \psi)$, $\mathcal{M}_\mathcal{P} \vdash (\Gamma \rightarrow \psi) \psi \rightarrow \gamma$, $\mathcal{M}_\mathcal{P} \vdash (\Gamma \rightarrow \psi) \Gamma \rightarrow \gamma$, $\mathcal{M}_\mathcal{P} \vdash \varphi \rightarrow [\alpha] \varphi$, $\mathcal{M}_\mathcal{P} \varphi \rightarrow [\alpha] \varphi$, $\forall \alpha \in \Pi$

6 AXIOMATIC DIAGRAMS OF RDL

Depending on the intended class of models for a particular RKB-ST different sets of axioms should be used. Next we show the three axioms representing the syntactic counterpart of the classes of models presented above.

- **DED:** $(i \rightarrow \varphi) \rightarrow (\Gamma \rightarrow \psi)(j \rightarrow \psi)$ (Deduction)
- **SEA:** $(\Gamma \rightarrow \psi) \psi \rightarrow \gamma$ for $\gamma \in \Phi_0$ and $\gamma \neq j \rightarrow \psi$ (Side-Eff ect Avoidance)
- **MON:** $\varphi \rightarrow [\alpha] \varphi \forall \alpha \in \Pi$ (Monotonicity)

As proposition 5.3 shows, DED is sound for $\mathcal{M}_\mathcal{P}$ DED and IND are sound with respect to $\mathcal{M}_\mathcal{P}$ DED and MON is sound with respect to $\mathcal{M}_\mathcal{P}$ and all of them are sound with respect to $\mathcal{M}_\mathcal{P}$. It will be a point for further research to check the following conjecture concerning the completeness of this set of axioms.

**Conjecture 6.1**

1) RDL$^1 = DDL + DED$ is complete w.r.t. $\mathcal{M}_\mathcal{P}$

2) RDL$^2 = DDL + DED + SEA$ is complete w.r.t. $\mathcal{M}_\mathcal{P}$

3) RDL$^3 = DDL + DED + MON$ is complete w.r.t. $\mathcal{M}_\mathcal{P}$

4) RDL$^4 = DDL + DED + SEA + MON$ is complete w.r.t. $\mathcal{M}_\mathcal{P}$

7 MAPPING RKB INTO RDL

In table 7.2 we sketch how one can map the components of a given RKB into RDL elements. Boldface components represent the formalization of the concepts of reflective architectures. Their representation in terms of RDL is presented in the third column.

In figure 7.3 four RKB-ST of different architectures are presented. BMS [11] whose goal is mainly to model default reasoning based on partial logics; DESIRE [12] that is intended to be a general architecture for problem solving based on a modular approximation; MIORD II [10] which is also a general architecture based on multi-valued logics and MC-Systems [5], a reflective architecture for theorem proving. Despite their different goals it is interesting to notice just looking at the figure 7.3 how similar they are, from the RDL perspective.

Three of the architectures contain a substrcuture that fits with the most basic reflective schema, shown in figure 7.1., that is an object and meta-level languages connected through a reflection upwards / reflection downwards mechanism.

![Figure 7.1 Object/Meta control flow.](image-url)
<table>
<thead>
<tr>
<th>REFLECTIVE KNOWLEDGE BASE</th>
<th>FORMALIZATION</th>
<th>RDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object and meta Level Languages</td>
<td>$RLA = (L, Δ, S, T)$ RKB-ST$= (RLA, U, M_L, M_Δ, Δ, B)$</td>
<td>$Φ_0$</td>
</tr>
<tr>
<td>Module/context and intra-unit inf. rules</td>
<td>$RLA = (L, Δ, S, T)$ RKB-ST$= (RLA, U, M_L, M_Δ, Δ, B)$</td>
<td>1) $Φ' \subseteq Φ$, formulas in RDL modelling inference rules 2) $Π^{\text{inn}}_0$</td>
</tr>
<tr>
<td>Reflection rules between units</td>
<td>$RLA = (L, Δ, S, T)$ RKB-ST$= (RLA, U, M_L, M_Δ, Δ, B)$</td>
<td>1) $Φ'' \subseteq Φ$, formulas in RDL modelling bridge rules 2) $Π^{\text{inter}}_0$</td>
</tr>
<tr>
<td>Domain knowledge</td>
<td>$RKB = (RKB-ST, MΣ, MΩ)$</td>
<td>$Φ''' \subseteq Φ$, formulas in RDL representing quoted formulas of units</td>
</tr>
<tr>
<td>Operational Semantics</td>
<td></td>
<td>1) $π_{\text{oper}} \in Π$, program in $Π$ representing the control flow 2) Axioms in RDL</td>
</tr>
<tr>
<td>Query, ?-q.</td>
<td></td>
<td>$Φ' \cup Φ'' \cup Φ''' \cup {φ</td>
</tr>
</tbody>
</table>

Table 7.2. Mapping between RKBs and RDL theories

To illustrate further the mapping between RKBs and RDL, let us recall example 3.4 and extend it with a particular toy domain knowledge. This example will clarify the main goal of our work, which is to be able to interpret executions in particular reflective architectures as proofs in the corresponding RDL theory.

**Example 7.1** The tweety example is an RKB over the RKB-ST, that is, RKB$\text{TWEETY}=(RKB$-ST$\text{BMS} \cup MΣ, MΩ)$, where:

- $MΣ(a) = \{(penguin \sqsupset \text{bird}, flies), (Prop), (penguin \rightarrow \text{Prop}, \text{bird} \rightarrow \text{Prop}, \text{flies} \rightarrow \text{Prop})\}$
- $MΩ(b) = \{\text{penguin} \sqsupset \text{bird}, \text{flies})\}$
- $MΩ(b) = \{\text{bird} \sqsupset \text{Constant}, \text{flies} \sqsupset \text{Constant}, \text{bird} \sqsupset \text{Constant}, \text{flies} \sqsupset \text{Constant}\}$ is a set of constants representing the propositions in $MΣ(a)$.
- $MΩ(b) = [\{penguin \sqsupset \text{bird}, \text{flies})\}$
- $MΩ(b) = [\{T(penguin) \& T(bird) \Rightarrow PA(\text{flies})\}$

This RKB is mapped into an RDL theory, called RTweety, accordingly to the following mapping (for any program $α \in Π$):

$$RTweety = \{$$

**Formulas representing Intra-Unit Inference rules:**

1. $(α(a - [φ \sqsupset ψ, l])) ∧ <α > (α - [φ, l])$
   $$\rightarrow <α^+_{aa} > (α - [φ, l])$$ (modus ponens in $a$)

2. $(α(b - [ψ])) ∧ (α(b - [φ]))$
   $$\rightarrow <α^+_{aa} > (b - [ψ])$$ (And Intro. in $b$)

3. $(α(b - [φ \sqsupset ψ])) ∧ <α > (b - [φ])$
   $$\rightarrow <α^+_{ba} > (b - [ψ])$$ (modus ponens in $b$)

**Formulas representing Bridge Rules:**

4. $(<α > (α - [φ, l]))) → <α^+_{aa} > (b - [T(φ)])**

5. $(<α > (α - [φ, l]))) → <α^+_{aa} > (b - [¬T(φ)])$

6. $(<α > (α - [φ, l]))) → <α^+_{aa} > (b - [¬T(φ)])$

7. $(<α > (α - [φ, l]))) → <α^+_{aa} > (b - [¬T(φ)])$

8. $(<α > (α - [φ, l]))) → <α^+_{aa} > (b - [¬T(φ)])$

9. $(<α > (α - [φ, l]))) → <α^+_{aa} > (b - [¬T(φ)])$

10. $(<α > (α - [φ, l]))) → <α^+_{aa} > (b - [¬T(φ)])$

To prove in RTweety that from knowing bird and not knowing penguin we get flies is equivalent to prove the next formula in RDL4:

$$RTweety \cup (a - [\{\text{bird}, \text{flies}\}), a - [\{\text{penguin}, u\}])$$

$$\vdash_{\text{RDL4}} \{\text{BMS- control}\}[u - [\{\text{flies}, l\}]]$$

which means that after the execution of the BMS system we get to a state in which the formula flies is true. The implicit control in BMS system, represented by the compound program $\text{BMS-control} = \{a^+_{aa} \sim a^+_{ab} \sim ba\}$, means that we make all possible deductions at unit $a$, we reflect up to unit $b$ all formulas in $a$, we make all possible deductions in unit $b$ and finally we reflect down to unit $a$ which can be proved, as the next proof tree shows. Proposition 4.4 is implicitly used in many steps. In the proof steps only the modus ponens in RDL is used.
Similarly, it can be proved that:

\[ \text{BMS} \]

\[ \text{DESIRE} \]

\[ \text{Figure 7.3 Comparison between reflective architectures.} \]

9 CONCLUSION

In this work we have focused on using dynamic logics to describe a particular case of computational dynamic programs called Reflective Knowledge-Bases Systems. These systems are built upon several languages, connected through Reification/Reflection mechanisms. Many Artificial Intelligence architectures embed such formalisms as the basic reasoning system. We have
presented a family of Dynamic Logics called Reflective Dynamic Logics that provides a general framework to
describe and compare meta-level architectures because
this allows to model the operational semantics of these
architectures in a common setting. This is the first
attempt to interpret executions in particular reflective
architectures as proofs in Dynamic Logic. Other
attempts are based on temporal semantics [13]. As a
next research step, we will try to show how several
reflective reasoning systems can be formalized in RDL.

10 ACKNOWLEDGMENTS

We are greatly indebted to Mara Manzano for many
useful comments made on a previous version of this
draft. Also Antonia Huertas, Wiebe van der Hoek, John
Jules Meyer, Fausto Giunchiglia and Luciano Serafini
commented on these ideas and helped us to improve
them.

This Research has been supported by the Spanish
research project Arrel, CICYT nº TIC92-0979, and the
Esprit III Basic Research Action nº 6156 DRUMS II.

REFERENCES

Mántaras R., Sierra C. (1994) : "Local multi-valued logics
in modular expert systems", Journal of Experimental and
Theoretical Artificial Intelligence, vol. 6, pp. 303-321.

"Combining Multiple-valued Logics in Modular Expert
Systems", in Proc. 7th Conference on Uncertainty in AI, Los
Angeles, Morgan Kaufman, p 17-25.

Object-centered representation language for knowledge
modelling. (submitted for publication).

KADS/(ML)2 model of a scheduling task", in J. Treur and
Th. Wetter eds Formal Specification of Complex Reasoning
Systems, Ellis Horwood Pub (in press).

"Multi-context systems as a specification framework for
complex reasoning systems", in J. Treur and Th. Wetter
eds Formal Specification of Complex Reasoning Systems, Ellis
Horwood Pub (in press).

Hierarchical Logics (or: How we can do without modal

with and between a Declarative Metatheory, JICAI-91, pp.
111-117.

F. Guenthner eds. Handbook of Philosophical Logic vol.2,

and Reflection, P. Maes and N. Nardi editors, North-
Holland

scheduling tasks in a reflective and modular architecture",
in J. Treur and Th. Wetter eds Formal Specification of

Architecture and philosophical applications, Ph. D.

modelling complex reasoning", Int. Journal of Intelligent

architectures for dynamic control, in: Proc. 4th Int
Workshop on Meta-Programming in Logic, META'94,
Pisa.

(1993) : "Comparing formal specification languages for
complex reasoning systems", in Formal Specification of
258-282, Ellis Horwood

Mechanized Formal Reasoning", Artificial Intelligence, Vol
13 No. 1,2, North Holland. Amsterdam.