A first step towards modeling semistructured data in hybrid multimodal logic

Nicole Bidoit* — **Serenella Cerrito**** — **Virginie Thion***

* *LRI UMR CNRS 8623, Université Paris 11, Centre d'Orsay.*

{bidoit,thion}@lri.fr

** *LaMI UMR CNRS 8042, Université d'Evry Val d'Essonne.*

{serena}@lami.univ-evry.fr

ABSTRACT. *XML documents and, more generally, semistructured data, can be seen as labelled graphs. In this paper we set a correspondence between such graphs and the models of a language of* hybrid multimodal logic. *This allows us to characterize a schema for semistructured data as a formula of hybrid multimodal logic, and instances of the schema (data graphs) as models of this formula. We also investigate how to express in such a logic integrity constraints on semistructured data, in particular some classes of constraints widely considered in the literature. The contribution of this work is twofold:*

1) We generalize the notion of schema, by proposing a definition of schema were references are "well typed" (contrary to what happens with DTDs).

2) We formalize semistructured data, schema for semistructured data and integrity constraints in a unique framework, namely hybrid multimodal logic.

KEYWORDS: *Database, Semistructured data, Integrity constraints, Schema, Multimodal and Hybrid logic.*

Introduction

In the database community the interest for *semistructured data* is more and more widespread. The history of such a notion has its roots in the popularity of XML as a standard for data exchange on the web [ABI 00].

Indeed, an XML document may be seen as a collection of data whose organization is "more flexible" with respect to, for instance, relational or object oriented databases. Why ? In absence of a grammar constraining data, as a DTD [RAY 01], for instance, an XML document can be seen as a logical organization of data which does not obey to any *a priori* constraint. But even when a DTD imposes some constraints on the organization of data, data which are "valid" with respect to the DTD still enjoy a

certain degree of freedom. For instance, the presence of the `?` operator in a DTD allows one to specify optional elements, the `ANY` operator allows one to leave the structure of a given element completely free, some XML attributes may be declared as optional, etc.

Several models for semistructured data, a notion which goes beyond XML, have been proposed [ABI 00]. In general, a semistructured database instance is seen as a labelled graph. In the literature, labels are sometimes attached to edges and sometimes to vertices. The labels are meant to convey information about some kind of logical data organization. In this paper, we opt for the first choice w.r.t. labels corresponding to XML tags. Moreover we attach data to leaves. Figure 1 is a classical example of semistructured data representing information about books and articles in a library.

It is well-known that the semantics of the family of *modal* logics is defined via graphs called *Kripke's models* [KRI 71]. Such logics provide a simple representation of graph properties and a rather powerful reasoning mechanism over such structures. The above description of semistructured data and of *modal* logic features leads us naturally to look at semistructured data as models of modal logics and vice versa. It motivates investigating the adequacy of modeling and reasoning over semistructured data by using modal logics. Indeed, we carry out this study by considering a variant of modal logics called *hybrid multimodal logic* [BLA 00]. The motivation behind choosing *hybrid multimodal logic* is based on the following observation [BLA 00]: although multimodal logic is a simple formalism for working with relational structures or multigraphs, it has no mechanism for referring to and reasoning about the individual nodes in such structures. Hybrid multimodal logic increases the effectiveness of modal logic as a representation formalism by allowing one to grasp the nodes of multigraphs via formulas. Roughly, this is achieved by mixing the modal logic ingredients with first order logic ingredients and allowing one to use node names and node variables as atomic formulas. As pointed out in [FRA 03], adding the ability to name graph nodes leads to violate the tree-like model property, which will be very useful in our framework: semistructured data (with references) are not trees but graphs.

Establishing the link between semistructured data and hybrid multimodal logic is a rather easy task. A pioneer work in this direction is [ALE 01] where a variant of propositional dynamic logic is used to formalize some constraints on semistructured data. More recently, [FRA 03] use hybrid logics to reason about semistructured data. Our work is complementary and our contribution is to propose a well founded formalization of schema for semistructured data allowing typed references and to provide a unique formalism for schema and constraints namely hybrid multimodal logic. Although the notion of semistructured data (data whose format is not fixed) and that of schema (a notion that imposes a fixed format on data) are contradictory, it may be useful to recall at this point the need for structure (more or less fixed) for data manipulation. The main problems can be presented by the two basic questions: (1) how do we express queries without knowing the structure of data ? (2) how do we optimize queries without knowing the structure of data ? Optimization is certainly the most important issue: more information about the logical and physical data organization

is available more efficient is the expected optimization. In this context, we are interested in logical organization and we would like to stress that it is composed by both the integrity constraints and the data format. Indeed when the schema can be considered as a set of integrity constraints it can be used not only to express queries but more importantly for optimizing queries [HAC 95, MCH 99, DEU 01a]. This paper first proposes a notion of semistructured data schema called *pattern grammar*. The main goal when defining pattern grammar is to introduce typed references which is not possible for DTDs [RAY 01, ABI 00]. We believe that it is very important to have a mechanism allowing one to specify, in the schema, the types of the referenced entities, so as to be able to enforce for example that `a child is a person`. As a matter of fact this example shows that the frontier between schema and integrity constraints is quite fuzzy.

The second part of this work is devoted to show that hybrid multimodal logic is expressive enough to formalize pattern grammars. In other word, we show that an hybrid multimodal formula can be associated to any pattern grammar in such a way that models of the formula and instances of the pattern grammar match. This result demonstrates that hybrid multimodal logic is the right logic for formalizing semistructured structured data (without format), integrity constraints over semistructured structured data and schema including typed references.

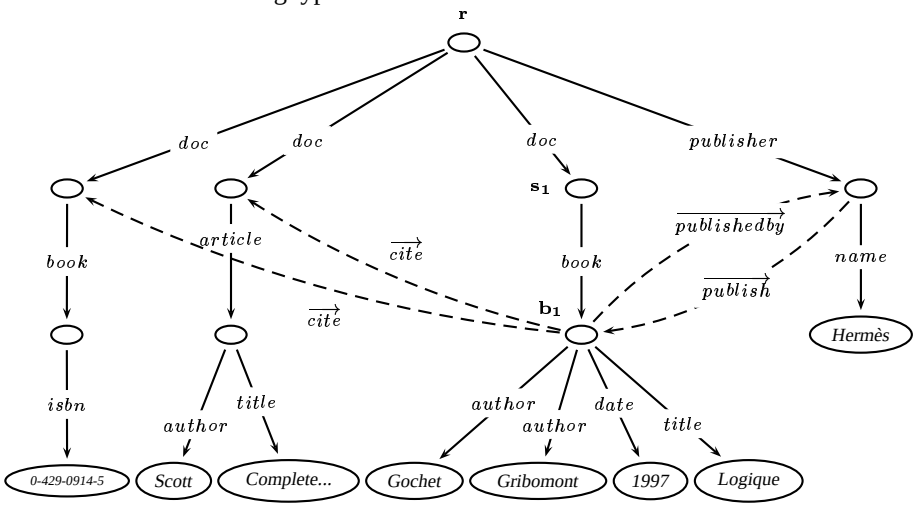


Figure 1. *Representation of a library database*

The paper is organized as follows. In the first section, we introduce semistructured data and hybrid multimodal logic. In the second section we show how to express some (popular) integrity constraints over semistructured data in such a logic. Next section is the core of the paper: the issue is to introduce pattern grammar, a notion of schema for semistructured data enforcing reference typing and then to show how to formalize pattern grammar in hybrid multimodal logic. We conclude the paper by comparing our approach with related works and by presenting some future work.

Proofs are outlined. They can be found in the technical report [BID 03].

1. Semistructured data and Hybrid multimodal logic

In this section we briefly present the notion of semistructured data, by giving its usual presentation, and we present the hybrid multimodal logic. Of course this section serves to draw an obvious link between semistructured data and model of the hybrid multimodal logic.

1.1. *Semistructured data*

A semistructured data is usually seen as a rooted, labelled, directed graph with valued leaves [ABI 97a, BUN 96]. This is the case for instance of the Object Exchange Model [ABI 97b].

The following definition of semistructured data used in our presentation is the one proposed in [BUN 97] up to some minor changes of notation and to the fact that nodes are labelled.

DEFINITION 1 (SEMISTRUCTURED DATA). — Let $\mathcal{E}$ be a set of edge labels and let $PROP$ be a set of node labels. A semistructured data (or a data graph) $\mathcal{S}$ is a rooted graph (S, r, R, V) where S is a finite set of nodes, r is the root of $\mathcal{S}$, $R = \{r_e \mid e \in \mathcal{E} \text{ and } r_e \in S \times S\}$ is a set of labelled edges between nodes of S[1] and V is a node labelling function $PROP \rightarrow Pow(S)$.

Figure 1 illustrates this definition. It is the classical example of semistructured data representing a library. This library contains information about documents (books and articles) and publishers. Following our definition, each label e in the graph, is associated to a binary relation r_e of $R = \{r_{doc}, r_{publisher}, r_{book}, r_{\overrightarrow{cite}}, \ldots\}$. The reader can notice that some of the edges of the data graph are represented by dashed arrows. These edges will be considered as references in the Section 3. Intuitively, references behave differently from other edges and in our example the edges labelled by $\overrightarrow{cite}$ serves to express that a document can quote other documents but do not serve to "create" a document. We would like also to emphasize that in general semistructured data do not support labels on nodes others than leaves. We choose not to enforce this restriction because it might be interesting to handle "data" on internal nodes for capturing XML non-referential attribute values.

This example (Figure 1) is used all along the presentation.

1. Each binary relation r_e gives the edges labelled by e in $\mathcal{S}$.

1.2. *Hybrid multimodal logic*

Modal propositional logics [ARE 99] are formalisms which are simple yet rather well suited to work with relational structures like graphs. Once several accessibility relations are considered, corresponding to different modal operators, one gets *multimodal logics*, whose models are essentially oriented graphs having different sorts of edges, or, which comes to the same, edges equipped with different labels.

However, traditional modal logics do not provide tools allowing one to make reference to specific vertices of a graph, which constitutes a limit on the expressive and reasoning power. For instance, it is not possible to express the fact that, for a given vertex s, there is no edge from s to s itself (irreflexivity) or that if s is linked to s' by an edge, then s' is also the source of an edge going back to s (symmetry). In general, standard multimodal logics do not suffice to express some interesting constraints on a data graph, as those introduced in [BUN 98] for instance, that we are going to study in Section 2, or to adequately formalize semistructured data schemas (see Section 3).

Hybrid modal logics [BLA 00] are extensions of modal logics which, even in their simplest forms, provide a mechanism to explicitly reference vertices in graphs. Since the aim of the present paper is not to fully present such logics, here we just recall some basic notions. We invite the interested reader to consult [ARE 99, BLA 95, ARE 01].

The language of hybrid modal logics allows one to name states and to express that a given formula holds at a given named state. This is made possible by four essential features:

Nominals which are special propositional symbols: each nominal "is true" at exactly one state of the model. Hence, a nominal "names" the unique state where it holds.

State variables which are, again, atomic formulas, enabling to denote states (under a given variable assignment).

Satisfaction operators of the form $@_u$ where u is either a state variable or a nominal. An expression $@_u\psi$, where ψ is a formula, says that ψ is satisfied at the state denoted or named by u.

The binder $\downarrow$ allows one to construct a formula of the form $\downarrow x\ \psi$, where x is a state variable and ψ is a formula. $\downarrow x\ \psi$ binds all the occurrences of x in ψ to the current state, that is the state where evaluation is occurring.

Modal hybrid logic is proof-theoretically and algorithmically "well behaved". In the present work, we investigate the relationship between semistructured data and multimodal hybrid logic. We leave the study of properties of proof systems for hybrid modal logic and of decidable fragments of such a logic, in relation to the formalization of semistructured data, to a future work.

In the next subsection we formally define the syntax and the semantics of the version of propositional hybrid multimodal logic that we use in the paper, and we suggest, via some examples, how to use it to modelize semistructured data.

1.3. *Syntax and semantics of hybrid multimodal logic*

The alphabet of the hybrid multimodal logic that we consider in this paper contains:

- a set of propositional symbols $PROP$ noted $p, q, ...$,
- the logical constant $\top$,
- a set of nominals $NOM = \{a, b, \cdots\}$,
- a set of state variables $SVAR = \{x, y, z, \cdots\}$,
- the binary connective $\wedge$,
- the unary connective $\neg$,
- a finite set of labels $\mathcal{E} = \{e_1, ..., e_n\}$,
- the modal unary operators $[e]$ where $e \in \mathcal{E}$,
- the binder $\downarrow$, and
- the operators $@_u$, where u is any state variable or nominal.

As usual, the sets $PROP$, NOM and $SVAR$ are pairwise disjoint.

It is clear from the definition of the alphabet above that, the "modal component" is provided by the usual operator [ ], the "multimodal component" is introduced by the set of labels (which leads to a set of modal operators $[e]$), and finally as explained just before, the hybrid features are provided by the introduction of variables, nominals and the hybrid operators $@_u$ and $\downarrow x$.

Well formed formulas are defined by:
WFF ::= $p \mid \top \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid [e]\psi \mid \langle e\rangle\psi \mid u \mid \downarrow x\ \psi \mid @_u\psi$
where ψ, ψ_1 and ψ_2 are well formed formulas, $p \in PROP$, $x \in SVAR$ and $u \in NOM \cup SVAR$.

In the following, we use also the operators $\vee$, $\Rightarrow$ and $\langle e\rangle$ classically defined by:
$\psi_1 \vee \psi_2 =_{def} \neg(\neg\psi_1 \wedge \neg\psi_2)$, $\psi_1 \Rightarrow \psi_2 =_{def} \neg\psi_1 \vee \psi_2$ and $\langle e\rangle\psi =_{def} \neg[e]\neg\psi$.
Intuitively, we recall that $[e]\varphi$ intends to check that φ holds in all states reachable through an edge labelled by e whereas $\langle e\rangle\varphi$ aims to verify that there exists a state reachable through an edge labelled by e where φ holds.

Examples of formulas are given just after the presentation of the semantics of hybrid multimodal logic.

Formally, a finite *model* $\mathfrak{M}$ of hybrid multimodal logic is a tuple $(S, s_0, R, V, \mathcal{I}_{nom})$ where:

- S is a finite set of *states* containing a distinguished element s_0;

• $R = \{r_e | e \in \mathcal{E}\}$ is a set of binary relations on S, called *accessibility relations* (R contains exactly one relation r_e for each label $e \in \mathcal{E}$);

• V is a function $PROP \rightarrow Pow(S)$, assigning to each proposition p the set of states where p holds;

• $\mathcal{I}_{nom}$ is a function $NOM \rightarrow S$ assigning a unique state to each nominal.

In the case of modal logics, different restrictions may be imposed on the accessibility relations, thereby obtaining different logics for which several styles of sound and complete proof systems exist (see, for instance, [FIT 83, WAL 90]). Here, such restrictions on accessibility relations are not considered and thus one can see "our" hybrid multimodal logic as an extension of multimodal logic K.

The semantics of the modal and hybrid operators is defined as follows.

A valuation (or variable assignment) is a function $g : SVAR \rightarrow S$ assigning a state to each state variable. The notation $g \overset{x}{\sim} g'$ (g' is a x-variant of g) means that g' and g are valuations (for a given $\mathfrak{M}$) such that g' is identical to g except for, possibly, the argument x.

As for modal logic, the evaluation of a formula with respect to a model needs to be done at some particular state of the model. The notion "$\mathfrak{M}$ satisfies ψ at state s w.r.t. a valuation g", noted $\mathfrak{M}, g, s \models_{hm} \psi$, is defined by induction on ψ as follows:

$\mathfrak{M}, g, s \models_{hm} \top$	for any $\mathfrak{M}, g, s$
$\mathfrak{M}, g, s \models_{hm} p$	iff $s \in V(p)$, where $p \in PROP$
$\mathfrak{M}, g, s \models_{hm} a$	iff $I_{nom}(a) = s$, where $a \in NOM$
$\mathfrak{M}, g, s \models_{hm} x$	iff $g(x) = s$, where $x \in SVAR$
$\mathfrak{M}, g, s \models_{hm} \psi_1 \wedge \psi_2$	iff $\mathfrak{M}, g, s \models_{hm} \psi_1$ and $\mathfrak{M}, g, s \models_{hm} \psi_2$
$\mathfrak{M}, g, s \models_{hm} \neg\psi$	iff $\mathfrak{M}, g, s \not\models_{hm} \psi$
$\mathfrak{M}, g, s \models_{hm} [e]\psi$	iff for any $s' \in S$, $(s, s') \in r_e$ implies $\mathfrak{M}, g, s' \models_{hm} \psi$
$\mathfrak{M}, g, s \models_{hm} \downarrow x\ \psi$	iff $\mathfrak{M}, g', s \models_{hm} \psi$ with $g \overset{x}{\sim} g'$ and $g'(x) = s$, where $x \in SVAR$
$\mathfrak{M}, g, s \models_{hm} @_x\psi$	iff $\mathfrak{M}, g, g(x) \models_{hm} \psi$ where $x \in SVAR$
$\mathfrak{M}, g, s \models_{hm} @_a\psi$	iff $\mathfrak{M}, g, I_{nom}(a) \models_{hm} \psi$ where $a \in NOM$

Intuitively, the formula $@_u\psi$ allows one to "jump" from the current state of evaluation to the state denoted by u and to evaluate ψ there. The formula $\downarrow x\ \psi$ as a side effect on the valuation g and modifies it: in the scope of ψ, the free variable x will denote the current state s.

The purpose of the following example is twofold: first of all, it shows how a semistructured data graph may be seen as a model of an hybrid multimodal language and then it serves to illustrate the semantics of the modal and hybrid operators $[e]$, $\langle e \rangle$, $@_u$ and $\downarrow x$.

EXAMPLE 2. — The language that we consider contains just a nominal, $root$, naming the root r of the graph (i.e. $\mathcal{I}_{nom}(root) = r$).

Now, the correspondence between a semistructured data graph $\mathcal{S}$ (Definition 1) and a

model $\mathfrak{M}$ of hybrid multimodal logic is obvious. We illustrate this correspondence on Figure 1:

- The nodes of the data graph $\mathcal{S}$ are the states of $\mathfrak{M}$.
- The root node r of the data graph $\mathcal{S}$ is the distinguished state s_0 of $\mathfrak{M}$.
- The labelled edges of $\mathcal{S}$ are the accessibility relations of $\mathfrak{M}$.
- Node labels used in the data graph $\mathcal{S}$ are the propositions of $\mathfrak{M}$ and of course the labelling of nodes in $\mathcal{S}$ corresponds to the assignement of propositions to states in $\mathfrak{M}$.

For instance, the presence of "Scott" on the leaf f of $\mathcal{S}$ corresponds in the model $\mathfrak{M}$ to the fact that $f \in V(Scott)$. Note that hybrid multimodal logic provides in a natural manner the labelling of internal nodes (states) by data.

Warning: for the sake of presentation, we sometimes give names to nodes (states) like r, s_1, b_1; these names should not be confused with node labels (propositions).

- The formula $[author]\neg Scott$ is satisfied at vertex b_1 because no state accessible from b_1 via an edge labelled by $author$ satisfies $Scott$. The formula $[publisher][name]Hermès$ is satisfied at the root r.
- The formula $\langle title\rangle\top$ is satisfied at b_1 because there is a vertex accessible from b_1 via an edge labelled by $title$.
- The formula $[doc](\langle book\rangle\top \vee \langle article\rangle\top)$ is satisfied at the root r.
- The formula $\downarrow x\ \langle\overrightarrow{publishedby}\rangle\langle\overrightarrow{publish}\rangle x$ is satisfied at b_1 because there is a state, accessible from b_1 via an edge labelled by $\overrightarrow{publishedby}$, which is the source of an edge labelled by $\overrightarrow{publish}$ going back to b_1. Let us note, in this example, the use of the binder $\downarrow$ in order to assign to the state variable x the present state of the evaluation, namely b_1. Note also that such a variable is used as a formula in the expression $\langle\overrightarrow{publish}\rangle x$.
- The formula $@_{root}[doc][article]\langle author\rangle\top$ is also satisfied at b_1, provided that the nominal $root$ names the root r of the data graph (which is the case because $I_{nom}(r) = root$): the operator $@_{root}$ allows one to "jump" from b_1 to r and to evaluate $[doc][article]\langle author\rangle\top$ at r.

The reader might now have some intuition about the ability of multimodal logic to express integrity constraints on semistructured data. For instance, the constraint "every article has at least one author" can be expressed by the formula $@_{root}[doc][article]\langle author\rangle\top$. Examples of constraints whose formalization exploits the presence of nominals and the hybrid operators are given later on.

In the following we also extend the language by adding two other modalities: the operator G and the operator F. The semantics of the operator G is: $\mathfrak{M}, s \models_{hm} G\psi$ iff for any $s' \in S$ such that $(s, s') \in R^+$, we have $\mathfrak{M}, g, s' \models_{hm} \psi$ where R^+ is the transitive closure of the relation $R = \bigcup_{e\in\mathcal{E}} r_e$. The formula $G\psi$ says that for any state s accessible via a path from the current state, ψ holds at s. The operator F is defined from G by $F\psi =_{def} \neg G\neg\psi$, similarly to what happens for $[e]$ with respect

to $\langle e \rangle$. The formula $F\psi$ says that there is a state s, accessible from the current one via a path of arbitrary length and labels, such that ψ is satisfied at s. Obviously, when such modalities are added, hybrid modal logic is no longer translatable into first order classical logic.

Below, an example illustrates the use of such new modalities to express properties of data graphs.

EXAMPLE 3 (CONTINUATION). — Again with respect to Figure 1, the formula $F\ Scott$ is satisfied at the root r because there is a state accessible from r where $Scott$ holds. But the formula $G[\overrightarrow{cite}]\langle book \rangle \top$ is not satisfied at r because b_1 is a state accessible from r where $[\overrightarrow{cite}]\langle book \rangle \top$ does not hold.

Recall that models of hybrid modal logic are unrestricted. The fact is that hybrid modal logic allows one to express such restrictions inside the logic itself that is by formulas. The paper [BLA 99] points out some hybrid formulas capable to express constraints on models which cannot be done in modal logic. For instance, the reflexivity of the accessibility relations r_e in a model can be expressed either by $\mathfrak{M} \models_{hm} \downarrow x\ \langle e \rangle x$ or by $\mathfrak{M} \models_{hm} @_x \langle e \rangle x$. As a matter of fact, this formula defines the class of models called T in modal logic. Another example is to express that all states in S are accessible from the root r via $\mathfrak{M} \models_{hm} \downarrow x\ @_{root} Fx$.

Remark 1 The language of the logic might also be easily extended with the set of inverse labels $\widetilde{\mathcal{E}} = \{\widetilde{e} \mid e \in \mathcal{E}\}$ and the operators $[\widetilde{e}]$, by setting $\mathfrak{M}, s \models_{hm} [\widetilde{e}]\psi$ iff for any $s' \in S$, $(s', s) \in r_e$ implies $\mathfrak{M}, s' \models_{hm} \psi$. Similarly, one might add an operator G^- for the inverse transitive closure of the union of accessibility relations, by setting $\mathfrak{M}, s \models_{hm} G^-\psi$ iff for any $s' \in S$ such that $(s', s) \in R^+$, we have $\mathfrak{M}, g, s' \models_{hm} \psi$. In such extended language it is possible also to follow graph paths "backwards".

The following semantical notions are needed:

- $\mathfrak{M}$ globally satisfies ψ w.r.t. a valuation g, noted $\mathfrak{M}, g \models_{hm} \psi$, iff for each state $s \in S$, we have $\mathfrak{M}, g, s \models_{hm} \psi$;
- $\mathfrak{M}$ globally satisfies ψ, noted $\mathfrak{M} \models_{hm} \psi$, iff for each valuation g, we have $\mathfrak{M}, g \models_{hm} \psi$.

2. Integrity constraints

Given that a semistructured data is a labelled (rooted) graph, or alternatively a model of hybrid multimodal logic, one can immediately think of expressing constraints over semistructured data by formulas of that logic. The topic of this section is to investigate integrity constraints in this framework. In the context of semistructured data, a schema can be seen as a special constraint on the general organization of the data graph. Thus, this section can be seen as a preliminary to the next one devoted to schemas for semistructured data and their formalization in hybrid multimodal logic.

Constraints on semistructured data are often expressed as properties of graph paths: these constraints are studied, for instance, in [ABI 97c, BUN 98, ALE 01, CAL 99, CAL 98, DEU 01b, NEV 03]. We consider the constraints studied in [BUN 98], which are there formalized by a sub-language of first order classical logic called $\mathcal{P}$; this class of constraints is a natural generalization of the constraints of [ABI 97c].

We first present the language $\mathcal{P}$. Then we exhibit some constraints which are expressible in $\mathcal{P}$ and we provide the translation in hybrid multimodal logic of these constraints. The section ends by establishing that hybrid multimodal logic expresses strictly more constraints than $\mathcal{P}$.

A given path in a graph is expressed in $\mathcal{P}$ by a first order classical formula with two free variables $\alpha(x, y)$ having one of the following forms:
- $x = y$, also written $\varepsilon(x, y)$ and denoting an *empty path*,
- $K(x, y)$, where K is a binary symbol,
- $\exists z(K(x, z) \wedge \beta(z, y))$, where K is a binary symbol and $\beta(z, y)$ is a formula expressing a path.

A formula of $\mathcal{P}$ is a classical first order formula having either the *forward* form $\forall xy(\alpha(root, x) \wedge \beta(x, y) \Rightarrow \gamma(x, y))$ or the *backward* form $\forall xy(\alpha(root, x) \wedge \beta(x, y) \Rightarrow \gamma(y, x))$ where $root$ is an individual constant and α, β, γ are formulas expressing paths. The path α is called the *prefix* of the constraint. Let us call here *path constraints* the constraints on paths which are expressible by formulas in $\mathcal{P}$. A path constraint having an empty prefix is called a *word constraint*.

In the following, the examples make use of the data graph represented in Figure 1. An example of path constraint is: "`Each document` y `cited by a book is a document accessible from the root of the database`". It is expressible in $\mathcal{P}$ via the formula (1):

$$\forall y(\exists v(r_{doc}(root, v) \wedge (\exists z(r_{book}(v, z) \wedge r_{\overrightarrow{cite}}(z, y)))) \Rightarrow r_{doc}(root, y))$$

This is a word constraint in the forward form.

These constraints say that if y is accessible from x via a path α then x is accessible from y via a path β (i.e. β is the converse of α). An example is the constraint "`given any book` x`, if` x `is published by` y `then` y `publishes` x". It is expressed in $\mathcal{P}$ by the formula (2):

$$\forall xy(\exists z(r_{doc}(root, z) \wedge r_{book}(z, x)) \wedge r_{\overrightarrow{publishedby}}(x, y) \Rightarrow r_{\overrightarrow{publish}}(y, x))$$

This is a path constraint in the backward form.

Path constraints are expressible in hybrid multimodal logic. For instance, (1) may be translated in hybrid multimodal logic as

$$@_{root}[doc][book][\overrightarrow{cite}]{\downarrow}x\ (@_{root}\langle doc\rangle x)$$

and (2) as

$$@_{root}[doc][book]{\downarrow}x\ ([\overrightarrow{publishedby}]\langle\overrightarrow{publish}\rangle x)$$

The reader may have noticed that the formula (1) is a particular word constraint so called extend constraints and the formula (2) is an example of inverse constraint. The *extend constraints* and the *inverse constraints* [BUN 98] are well-known and helpful classes of constraints.

Let $\mathcal{H}$ be the sub-language of hybrid multimodal logic which does not contain the operators F and G. *E*ach formula ψ of $\mathcal{P}$ has a translation $\tau(\psi)$ into $\mathcal{H}$. We show:

Theorem 1 Let $\mathfrak{M}$ be a model of hybrid multimodal logic and $\mathcal{I}$ be the corresponding first order interpretation [2]. There is a translation τ of formulas of $\mathcal{P}$ into formulas of hybrid multimodal logic $\mathcal{H}$ such that, for any formula ψ of $\mathcal{P}$,

$$\mathcal{I} \models_{fo} \psi \text{ iff } \mathfrak{M} \models_{\mathrm{hm}} \tau(\psi)$$

This result shows that hybrid multimodal logic is at least as expressive as $\mathcal{P}$. This property can be directly proved, as in [BID 03], or else inferred from a result in [ARE 99] showing that $\mathcal{H}$ as exactly the same expressive power as the fragment of classical first order logic said "bounded", which includes $\mathcal{P}$.

Note that there are natural constraints on paths in a data graph which cannot be expressed in $\mathcal{P}$, for instance: `a book has exactly one isbn number`. Such a constraint is easily expressible in $\mathcal{H}$, thanks to the presence of the binder $\downarrow$:

$$@_{root}[doc][book]{\downarrow}x\ (\langle isbn\rangle{\downarrow}y\ (@_x[isbn]y))$$

This shows:

Theorem 2 Hybrid multimodal logic $\mathcal{H}$ is strictly more expressive than $\mathcal{P}$.

Moreover, it is well known that no constraint involving the transitive closure of a relation is expressible in first order classical logic. For instance, under the hypothesis that documents may appear at any depth in the data graph, the constraint "`The database contains a document whose author is Scott`" is not formalizable in first order logic, hence is not formalizable in $\mathcal{P}$. But the full language of hybrid multimodal logic used in this paper contains the transitive closure operators G and F. This allows us to formalize the above constraint via the hybrid multimodal logic formula $\langle author\rangle Scott \vee (F\langle author\rangle Scott)$.

Since the implication problem for $\mathcal{P}$ is undecidable, the authors of [BUN 98] also address the issue of fragments of $\mathcal{P}$ decidable for such problem. The satisfiability problem for hybrid multimodal logic (where "ϕ is satisfiable" means: there is a model $\mathfrak{M}$, a state s and a valuation g such that $\mathfrak{M}, s, g \models \phi$) is also undecidable [ARE 99]. Hence, it would be useful to study decidable fragments of hybrid multimodal logic.

2. We do not detail here the correspondence between $\mathfrak{M}$ and $\mathcal{I}$, which is straightforward: essentially, accessibility relations are translated by relations interpreting binary predicates.

Because the undecidability is partly due to the presence of $\downarrow$, a challenging issue is the characterization of decidable fragments of hybrid multimodal logic containing transitive closure operators and, possibly, some restricted forms of $\downarrow$. For course, the challenge would be to exhibit decidable fragments powerful enough to express interesting integrity constraints over semistructured data [3]. This will be the subject of future work.

However, it is known that the class of *word constraints* is a subset of path constraints for which the implication problem is known to be decidable [ABI 97c]. This kind of constraints can easily be formalized in the decidable fragment of $\mathcal{H}$ not containing $\downarrow$, provided that one considers a language equipped with inverse labels, as in Remark 1.

Let consider a forward word constraint $\forall x(\beta(r,x) \rightarrow \gamma(r,x))$ where $\beta(r,x)$ corresponds to the path $b_1, \cdots, b_m$ and $\gamma(r,x)$ corresponds to the path $g_1, ..., g_n$. Such a constraint is expressible by a formula of $\mathcal{H}$: $@_{root}[b_1] \cdots [b_m]\langle\widetilde{g_n}\rangle \cdots \langle\widetilde{g_1}\rangle root$. Similarly, a backward word constraint $\forall x(\beta(r,x) \rightarrow \gamma(x,r))$ is expressible by $@_{root}$ $[b_1] \cdots [b_m]\langle g_n\rangle \cdots \langle g_1\rangle root$. Indeed, the use of inverse labels has been suggested in [ALE 01].

3. Schemas for semistructured data

The aim of this section is to study the notion of schema for semistructured data.

As we have already pointed out in the introduction, having a well defined notion of schema for semistructured data is quite important, because it increases the efficiency of query semantical optimization and thus the efficiency of query evaluation. Since semistructured data contain references, an adequate notion of schema should provide a mechanism for specifying the "types" of the referenced entities. This is not possible for the notions of schema currently found in the literature on XML, as DTD [RAY 01, ABI 00], XML-Schema [XML], DSD [KLA 00], DCD [DCD].

Regular expressions types [HOS 00], *regular expressions pattern matching* [HOS 03], *tree grammars* and *tree automata*[4][Dal 03] are other features allowing to specify a schema. However, all these formalisms assume that semistructured data are trees.

In this section, we propose a notion of schema which is more general. The main contribution is a treatment of reference declarations ensuring to well type referenced data. To this purpose, we define a schema as a particular graph grammar called *pattern grammar*. Then, we focus on the specification of such a schema via a formula of hybrid multimodal logic. The problem solved here is: given a pattern grammar (or schema) $\mathcal{G}$ and a data graph $\mathfrak{M}$, is there a formula ψ such that $\mathfrak{M}$ is an instance of $\mathcal{G}$ if and only if $\mathfrak{M}$ satisfies ψ?

3. A decidability result for a restricted form of $\downarrow x$ in $\mathcal{H}$ is given in [MAR 02].
4. A tree automaton can also be seen as a schema since its denotation can be defi ned as the set of trees accepted by the automaton.

3.1. *Schemas for semistructured data and well-formed references*

Since a schema will be a particular grammar, we first define what we mean here by *grammar*. To this purpose, we first define the notions of *pattern* and *rule*. We use:

• a finite set $\mathcal{V}$ of symbols called *n*on-terminal symbols, containing at least the symbol $Root$,

• a finite set of labels $\mathcal{E}$ disjoint from $\mathcal{V}$ and partitioned in two subsets E and $\overrightarrow{E}$: labels in $\overrightarrow{E}$ are called *references*, and

• a particular symbol Λ (empty word).

By convention, a non terminal symbol is a word starting with a capital letter while a label starts with a non-capital letter. References are overlined by the symbol $\rightarrow$.

The first notion that we need is the notion of pattern:

DEFINITION 4 (PATTERN). — A *pattern* $expr$ is an expression having one of the following forms:

1) Λ is the *empty pattern*;

2) $(e\ N)^{op}$ is an *elementary pattern* when $e \in \mathcal{E}$, $N \in \mathcal{V}$ and $op \in \{*, +, !, ?\}$;

3) $m_1, \cdots, m_k$ is a *conjunctive pattern* when for each $j \in [1..k]$, m_j is an elementary pattern;

4) $m_1 \mid \cdots \mid m_k$ is a *disjunctive pattern* when for each $j \in [1..k]$, m_j is a conjunctive pattern;

EXAMPLE 5. — Let $\mathcal{E}_1 = E_1 \cup \overrightarrow{E}_1$ be a set of labels where the labels are $E_1 = \{doc, publisher, name, article, book, author, title, date, isbn\}$, and the references are $\overrightarrow{E}_1 = \{\overrightarrow{publish}, \overrightarrow{cite}, \overrightarrow{publishedby}\}$.
Let $\mathcal{V}_1 = \{Root, Publisher, Doc, Art, Book, Name, Dat, Isb\}$ be a set of non terminal symbols.
Below, some examples of patterns are proposed:

1) $(doc\ Doc)^*$ is an elementary pattern. Intuitively, if this pattern "matches" a data graph at a state s, then zero or more edges labelled by *doc* are outgoing from s.

The patterns $(doc\ Doc)^+$, $(doc\ Doc)^!$ and $(doc\ Doc)^?$ are variants of $(doc\ Doc)^*$ able to express cardinality constraints on the number of edges having source s (respectively: at least 1, exactly 1, zero or 1).

2) $(author\ Nom)^+, (date\ Dat)^!, (\overrightarrow{cite}\ Doc)^*$ is a conjunctive pattern expressing that at least one edge labelled $author$, exactly one edge labelled by $date$ and, possibly, some edges labelled by the reference $\overrightarrow{cite}$ must come out of state s.

3) $(article\ Art)^! \mid (book\ Book)^!$ is a disjunctive pattern expressing that either exactly an edge labelled by $article$ or else exactly an edge labelled by $book$ must come out of state s.

4) $((author\ Nom)^{+} \mid (isbn\ Isb)^{!}), (\overrightarrow{cite}\ Doc)^{*}$ is not a legal pattern, according to our definition, however what it intuitively expresses is captured by the legal pattern : $(author\ Nom)^{+}, (\overrightarrow{cite}\ Doc)^{*} \mid (isbn\ Isb)^{!}, (\overrightarrow{cite}\ Doc)^{*}$.

A *pattern grammar* is defined as a set of rules, where rules are defined as:

DEFINITION 6 (RULE). — A rule is an expression having the form $N ::= expr$ where N is a non terminal symbol and $expr$ is a pattern.

EXAMPLE 7 (CONTINUATION). — Below, a set of rules is presented, some of which use the patterns of Example 5 :

$\mathcal{R}_1 = \{$

$$
\begin{array}{rcl}
Root & ::= & (doc\ Doc)^{*}, (publisher\ Publisher)^{*} \\
Publisher & ::= & (name\ Name)^{!}, (\overrightarrow{publish}\ Book)^{*} \\
Doc & ::= & (article\ Art)^{!} \mid (book\ Book)^{!} \\
Art & ::= & (author\ Name)^{+}, (title\ Name)^{!}, (date\ Dat)^{?}, (\overrightarrow{cite}\ Doc)^{*} \\
Book & ::= & (isbn\ Isb)^{!}, (\overrightarrow{cite}\ Doc)^{*} \mid (author\ Name)^{+}, (date\ Dat)^{!}, \\
 & & (title\ Name)^{!}, (\overrightarrow{cite}\ Doc)^{*}, (\overrightarrow{publishedby}\ Publisher)^{!} \\
Name & ::= & \Lambda \quad Dat ::= \Lambda \quad Isb ::= \Lambda\ \}
\end{array}
$$

A pattern grammar is, as we announced, a set of rules obeying to some restrictions:

DEFINITION 8 (PATTERN GRAMMAR). — A *pattern grammar* $\mathcal{G}$ is a tuple $(\mathcal{V}, Root, \mathcal{E}, \mathcal{R})$ where

1) $Root$ is the start symbol of the grammar and for any e and *op*, no elementary pattern $(e\ Root)^{op}$ occurs in $\mathcal{R}$.

2) the set of rules $\mathcal{R}$ contains exactly one rule for each non terminal symbol in $\mathcal{V}$,

3) for each couple of elementary patterns $(e_1\ N_1)^{op_1}$ and $(e_2\ N_2)^{op_2}$ occurring in $\mathcal{R}$, we have $e_1 = e_2$ implies $N_1 = N_2$,

EXAMPLE 9 (CONTINUATION). — The rule set $\mathcal{R}_1$, together with the set of non terminal symbols $\mathcal{V}_1$ and the set of labels $\mathcal{E} = E_1 \cup \overrightarrow{E}_1$, constitute a pattern grammar $(\mathcal{V}_1, Root, \mathcal{E}_1, \mathcal{R}_1)$, called $\mathcal{G}_1$ in the following. Note that there is a rule for each symbol in $\mathcal{V}_1$ and that the label $date$ is used in two patterns having operators corresponding to different cardinality constraints: $(date\ Dat)^{!}$ and $(date\ Dat)^{?}$. Note also that different labels are associated to the same non terminal symbol, for instance $(author\ Name)^{+}$ and $(title\ Name)^{!}$.

If the rule associated to Art was changed to $Art ::= (title\ Name)^{!}, (\overrightarrow{cite}\ Book)^{*}$, then $\mathcal{R}_1$ would contain both a pattern $(\overrightarrow{cite}\ Book)^{*}$ in the rule for Art and a pattern $(\overrightarrow{cite}\ Doc)^{*}$ in the rule for $Book$, which is forbidden by the condition (3) of the definition of pattern grammar (Definition 8).

More generally, note that forbidding the presence of two rules $N ::= expr_1$ and $N ::= expr_2$ having the same left side in a grammar is without loss of generality,

because such rules may be expressed by the unique rule $N ::= expr_1 \mid expr_2$ (as it is usually done in formal grammars).

Notations: Given $e \in \mathcal{E}$, $Symb(e)$ is the unique non terminal symbol $N \in \mathcal{V}$ such that $(e\ N)^{op}$ (for any *op*) is a pattern occurring in $\mathcal{R}$. Given any non terminal symbol $N \in \mathcal{V}$, $Label(N)$ is the set of labels e in $\mathcal{E}$ such that $Symb(e) = N$. For instance, for $\mathcal{G}_1$, $Symb(\overrightarrow{cite})$=*Doc*, $Symb(date)$=Dat, $Label(Dat) = \{date\}$ and $Label(Name) = \{author, title, name\}$.
$Expr(N)$ denotes the right-hand side of the rule associated to the non terminal symbol N in the grammar. For instance, for the grammar $\mathcal{G}_1$, $Expr(Root)$ is the pattern $(doc\ Doc)^*, (publisher\ Publisher)^*$ and $Expr(Name)$ is the empty pattern Λ.

It is easy to associate to a pattern grammar $\mathcal{G}$ a dependency graph. Its vertices are the non terminal symbols and there is an edge from N to M when M occurs in an elementary pattern $(e\ M)^{op}$ in $Expr(N)$ and $e \in E$. We emphasize that the dependency graph is built ignoring elementary patterns using a reference (a label in $\overrightarrow{E}$).

In the following, we say that a non terminal symbol N *is accessible from* $Root$ in the grammar G if and only if there is a path going from $Root$ to N in the dependency graph for G.

EXAMPLE 10 (CONTINUATION). — The dependency graph associated to the grammar $\mathcal{G}_1$ of Example 9 is drawn in Figure 2. In this graph, we note that the vertex Doc is not the target of any edge issued from $Book$ because there is no elementary pattern $(e\ M)^{op}$ with $e \in E$ in $Expr(Book)$. The expression $Expr(Book)$ does contain the elementary pattern $(\overrightarrow{cite}\ Doc)^*$, however, since $\overrightarrow{cite}$ belongs to $\overrightarrow{E}$, this pattern does not induce an edge from $Book$ to Doc.

We are now ready to define a schema for semistructured data as a particular kind of pattern grammar:

DEFINITION 11 (SCHEMA). — A schema for semistructured data is a pattern grammar $\mathcal{G} = (\mathcal{V}, Root, E \cup \overrightarrow{E}, \mathcal{R})$ such that each non terminal symbol of $\mathcal{V}$ is accessible from $Root$ in G.

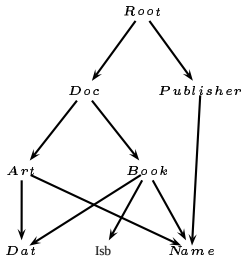


Figure 2. *Dependency graph associated to grammar* $\mathcal{G}_1$

EXAMPLE 12 (CONTINUATION). — The pattern grammar $\mathcal{G}_1$ is a data schema: all the vertices of its dependency graph (shown in Figure 2) are accessible from $Root$. On

the other hand, let us consider the grammar $\mathcal{G}_2 = (\mathcal{V}_1, Root, \mathcal{E}_1, \mathcal{R}_2)$ where the rules are the same as those in $\mathcal{R}_2$ except for the rule associated to Doc, which is changed to Doc::=$(book\ Book)^!$. The dependency graph $\mathcal{G}_2$ is shown in Figure 3. The vertex associated to the non terminal symbol Art is not accessible from $Root$, therefore the grammar $\mathcal{G}_2$ is not a schema.

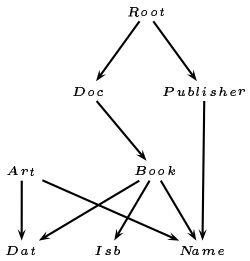


Figure 3. *Dependency graph associated to grammar $\mathcal{G}_2$*

A pattern grammar, similarly to DTD's, allows the specification of *recursive* data structures, as, for instance, sequences, trees, etc. The example 13 illustrates this.

EXAMPLE 13 (RECURSIVE SCHEMA). — Consider the grammar $\mathcal{G}_3 = (\mathcal{V}_3, Root, \mathcal{E}_3, \mathcal{R}_3)$ where $\mathcal{V}_3 = \{Root, Tr, L\}$, $\mathcal{E}_3 = \{tree, lt, rt, leaf\}$ and

$$R_3 = \{ \quad \begin{array}{lll} \text{Root} & ::= & (tree\ Tr)^+ \\ \text{Tr} & ::= & (lt\ Tr)!, (rt\ Tr)! \\ & & \mid (leaf\ L)! \quad \} \end{array}$$

The pattern grammar $\mathcal{G}_3$ is a recursive schema specifying binary trees.

In the following, we use indifferently the expressions "pattern grammar" and "schema" (obviously assuming that the considered grammar is indeed a schema).

In order to define the notion of instance of a schema, we proceed in two steps: intuitively, the first one sets the conditions under which a data graph "matches" a pattern grammar and lays the grounds for "type checking" references; the second one essentially checks that references are well typed.

In the following,

- the schema considered is a pattern grammar $\mathcal{G} = (\mathcal{V}, Root, E \cup \overrightarrow{E}, \mathcal{R})$;
- the data graph $\mathcal{S}$ is a semistructured data (S, r, R, V) (Definition 1); we note $Out_e(s)$ the set of states s_1 such that $(s, s_1) \in r_e$;
- if U is a set of states, the expression $|U| \cong card(op)$ means: $|U| \geq 0$ when $op = *$, $|U| > 0$ when $op = +$, $|U| = 1$ when $op =!$ and $0 \leq |U| \leq 1$ when $op =?$;
- the data subgraph obtained from $\mathcal{S}$ by erasing all references is noted $Pre(\mathcal{S})$ and is formally defined as $(S, r, Pre(R), V)$, where $Pre(R) = \{r_e | r_e \in R\ et\ e \in E\}$.

DEFINITION 14 (MATCHING AND MARKING). — Suppose that $Pre(\mathcal{S})$ is acyclic. The relations **matches** and **strictly matches**, over patterns, states and expressions, are defined by mutual recursion as follows.

A pattern $expr$ **strictly matches** $\mathcal{S}$ at state s if and only if $expr$ matches $\mathcal{S}$ at s and, for any $l \in \mathcal{E} - E_{expr}$, where E_{expr} is the set of labels occurring in $expr$, $Out_l(s) = \emptyset$.

Two **marking functions** $mark$ and req, assigning a set of non terminal symbols to a state of S, are recursively defined at the same time as the relation matches. Initially, for any state s, we set $mark(s) = \emptyset$ if s is not r, $mark(r) = \{Root\}$, and, for any state s, $req(s) = \emptyset$.

The relation "pattern $expr$ **matches** $\mathcal{S}$ at a state s of $\mathcal{S}$" is defined as follows:

1) $(e\ N)^{op}$ matches $\mathcal{S}$ at s iff

in the case where $e \in E$ we have: $|Out_e(s)| \cong card(op)$ and, for any $s_1 \in Out_e(s)$, the pattern $Expr(N)$ strictly matches $\mathcal{S}$ at s_1. For any $s_1 \in Out_e(s)$, the new value of $mark(s_1)$ is defined by adding N to the old value of $mark(s_1)$.

in the case where $e \in \overrightarrow{E}$, we have: $|Out_e(s)| \cong card(op)$ and, for any $s_1 \in Out_e(s)$, the new value of $req(s_1)$ is defined by adding N to the old value of $req(s_1)$.

2) Λ matches $\mathcal{S}$ at s iff, for each $e \in \mathcal{E}$, $Out_e(s) = \emptyset$.

3) $(e_1\ N_1)^{op_1}, \cdots, (e_j\ N_j)^{op_j}$ matches $\mathcal{S}$ at s iff for any k in $[1..j]$, the pattern $(e_k\ N_k)^{op_k}$ matches $\mathcal{S}$ at s.

4) $expr_1 | \cdots | expr_n$ matches $\mathcal{S}$ at s iff there is $i \in [1..n]$ such that the pattern $expr_i$ strictly matches $\mathcal{S}$ at s.

Remark 2 If the pattern $(article\ Art)^! \mid (book\ Book)^!$ matches $\mathcal{S}$ at s then s has exactly one outgoing edge labelled either by $article$, or by $book$: it can not have two outgoing edges respectively labelled by $article$ and $book$. This is enforced by the definition of matching for a disjunctive pattern $expr_1 | \cdots | expr_n$ which requires that there is $i \in [1..n]$ such that pattern $expr_i$ **strictly** matches $\mathcal{S}$ at s. In fact,

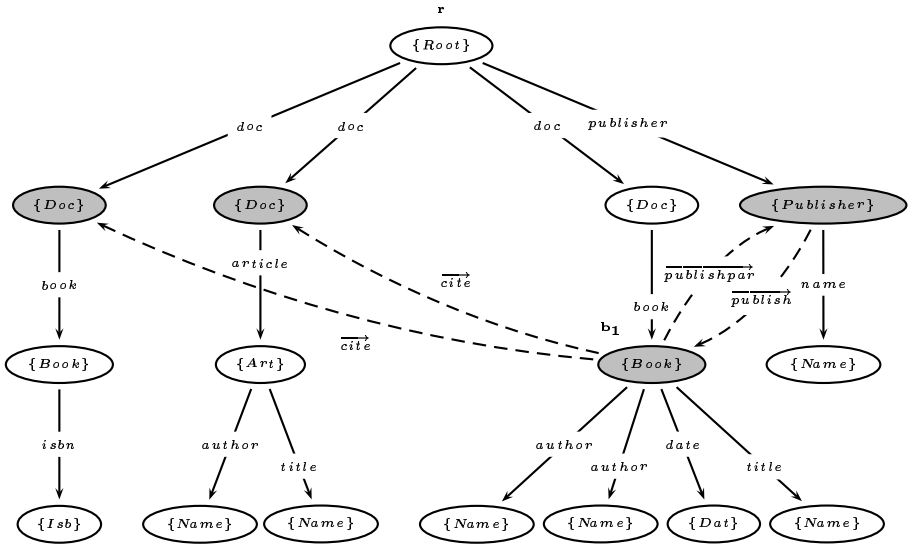
- either $(article\ Art)^!$ strictly matches $\mathcal{S}$ at s which implies that no edge labelled by $e \in \mathcal{E} - \{article\}$ may have s as a source,
- or $(book\ Book)^!$ strictly matches $\mathcal{S}$ at s which implies that no edge labelled by $e \in \mathcal{E} - \{book\}$ may have s as a source.

The acyclicity condition on $Pre(\mathcal{S})$ guarantees that matching is well defined. The above definition might be generalized and such a restriction on $Pre(\mathcal{S})$ might be avoided, modulo some technical modification. However, since the definition of instance given below requires $Pre(\mathcal{S})$ to be a tree, we have chosen to define matching for acyclic graphs.

DEFINITION 15 (INSTANCE OF A SCHEMA). — A data graph $\mathcal{S}$ is an instance of a schema $\mathcal{G}$ when the following conditions are satisfied:
(i) $Pre(\mathcal{S})$ is an (oriented) tree having root r,
(ii) $Expr(Root)$ strictly matches $\mathcal{S}$ at the root r, and
(iii) For any state s of S, if $req(s) \neq \emptyset$ then $req(s) = mark(s)$.

EXAMPLE 16 (CONTINUATION). — The data graph of Figure 1 is an instance of the schema $\mathcal{G}_1$. Here, $Pre(\mathcal{S})$ is the graph obtained from the figure by erasing the

dashed edges (the references). Figure 4 shows the values of the functions $mark$ and req for this graph with respect to the pattern grammar $\mathcal{G}_1$. The values of the function $mark$ appear in the appropriate nodes of Figure 1. The data graph of Figure 1 is an instance of the pattern grammar $\mathcal{G}_1$ because for any vertex the value of req, when it is non-empty, is equal to the value of $mark$. In the figure, we represent by grey nodes the vertices for which the function $req()$ returns a non empty value. Note that such vertices are the targets of references, as required by the definition of the function req.



Caption: {Doc} means that, in this vertex, $mark = \{Doc\}$ and $req = \emptyset$

{Doc} means that, in this vertex, $mark = required = \{Doc\}$.

Figure 4. *Values of the functions $mark$ and req for the data graph of Figure 1 w.r.t. grammar $\mathcal{G}_1$*

The following properties clarify the intended meaning of the functions $mark$ and req, whose intuitive role is to allow one to check the type of the reference vertices. The first property depends on the tree structure imposed on $Pre(\mathcal{S})$.

Property 1 If $Pre(\mathcal{S})$ is an (oriented) tree rooted at r and $Expr(Root)$ matches the data graph $\mathcal{S}$ at root r, then, for any vertex s, $mark(s)$ is a singleton, i.e. vertex s is marked by exactly one non terminal.

Sketch of proof : Let $Pre(\mathcal{S})$ be an oriented tree rooted at r and let assume that $Expr(Root)$ strictly matches the data graph $\mathcal{S}$ at root r. Let s be a vertex of S. Property 1 is proved in two steps:

1) We prove that there is a pattern $expr$ (in $\mathcal{G}$) such that $expr$ strictly matches $\mathcal{S}$ at s and that $mark(s)$ contains at least one element: This is proved by induction on the size of the path going from r to s in $Pre(\mathcal{S})$.

2) We prove that $mark(s)$ contains at most one element: It is is proved by *reductio ad absurdum* by exploiting the fact that $Pre(\mathcal{S})$ is a tree. □

The second property is a consequence of the condition (ii) on the rules of a pattern grammar, which has enabled us to define $Symb(e)$ as the unique non terminal symbol occurring in elementary patterns associated to a label e.

Property 2 If $Pre(\mathcal{S})$ is an (oriented) tree rooted at r and $Expr(Root)$ strictly matches the data graph $\mathcal{S}$ at root r, then, for any vertex s of $\bigcup_{s_1 \in S} out_e(s_1)$, we have $mark(s) = \{Symb(e)\}$.

The proof of Property 2 uses the first step of Property 1 and the definition of $Symb$. This property allows us to deduce that if $Pre(\mathcal{S})$ is a tree rooted at r and $Expr(Root)$ strictly matches the data graph $\mathcal{S}$ at root r, then an edge having a vertex s as target and e as label determines the value of the function $mark$ for this vertex, and this value is $\{Symb(e)\}$. Hence, $Symb(e)$ corresponds to a "type" for s.

EXAMPLE 17 (CONTINUATION). — Let us consider Figure 4. An edge labelled by $book$ has vertex b_1 as target, therefore we can infer that b_1 "has type" $Symb(book)$, that is $Book$. In order to check that b_1 has indeed type $Book$, it suffices to check that the edges outgoing from this vertex verify the rule associated to $Book$ in $\mathcal{G}$.

Observe that the notion of schema that we propose allows one to specify that, for any instance of the schema, any reference edge must be "well typed": the target of a reference has a value for the function req which is determined by the schema. Formally, this corresponds to conditions (ii) and (iii) in Definition 15. *This typing constraint on references is not expressed by other notions of schema for semistructured data in the literature, typically the DTD.* For instance, in a DTD, one could not say that the reference link $\overrightarrow{publishedby}$ must point on a publisher (rather than on a book, an article, etc).

3.2. *Schema for semistructured data and hybrid multimodal logic*

In this section we are going to show that a schema for semistructured data (as defined in Subsection 3.1) can be expressed by a formula of hybrid multimodal logic, just like integrity constraints are.

Our aim is to define a translation function $TRANS$ associating to any schema $\mathcal{G}$ a formula $TRANS(\mathcal{G})$ of hybrid multimodal logic such that: for any semistructured data graph $\mathfrak{M}$, $\mathfrak{M}$ is an instance of $\mathcal{G}$ if and only if $\mathfrak{M}$ (globally) satisfies $TRANS(\mathcal{G})$. This classical translation process is highlighted by the diagram of Figure 5.

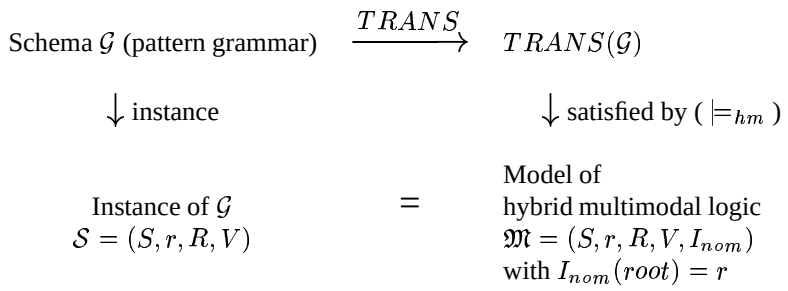


Figure 5. *Principle of schema translation*

Since the structures $\mathfrak{M}$ and $\mathcal{S}$ are essentially the same, in the following we identify them and we note both $\mathfrak{M}$.

The definition of an instance of a schema states a condition, namely $Pre(\mathfrak{M})$ is a tree, which is independent of the pattern grammar specifying the schema. We start by translating this condition by a formula of hybrid multimodal logic noted $tree$.

The fact that $Pre(\mathfrak{M})$ is the sub-graph of $\mathfrak{M}$ obtained by erasing those edges whose labels are references is translated, in the formula of $tree$, thanks to:

- the use of the modal operators $\langle e \rangle$ and $[e]$ restricted to labels $e \in E$, and
- a similar restricted use of the operators F and G:

Let $R^E = \bigcup_{e \in E} r_e$ (the set of the accessibility relations associated to labels which are not references) and let R^{E+} be the transitive closure of R^E. Let G^E be the operator whose semantics is defined by $\mathfrak{M}, g, s \models_{hm} G^E\varphi$ iff for any $s' \in S$ such that $(s, s') \in R^{E+}$, it is true that $\mathfrak{M}, g, s' \models_{hm} \varphi$. In the same way as F can be defined from G, the operator F^E is defined by $F^E\psi =_{def} \neg G^E \neg \psi$.

For the sake of readability, we use in the following the shorthands G^{E*} and F^{E*} respectively defined by $G^{E*}\psi =_{def} \psi \wedge G^E\psi$ and $F^{E*}\psi =_{def} \psi \vee F^E\psi$; they correspond to the transitive *and reflexive* closure of R^E. Similarly, we use G^* and F^* respectively defined by $G^*\psi =_{def} \psi \wedge G\psi$ and $F^*\psi =_{def} \psi \vee F\psi$.

$Pre(\mathfrak{M})$ is a tree if and only if the following conditions are satisfied:

1) all the states of $Pre(\mathfrak{M})$ are accessible from $root$, which is translated in hybrid multimodal logic by the formula: $\psi_1 =_{def} \downarrow x \, @_{root} F^{E*} x$;

2) $Pre(\mathfrak{M})$ does not contain cycles, translated in hybrid multimodal logic by the formula $\psi_2 =_{def} \downarrow x \, \neg F^E x$;

3) each state of $Pre(\mathfrak{M})$ has at most one predecessor, translated in hybrid multimodal logic by the formula $\psi_3 =_{def}$
$@_{root} G^{E*} \downarrow x \; @_{root} G^{E*} \downarrow y \; @_{root} G^{E*} \downarrow z \; (\bigvee_{e \in E} @_y \langle e \rangle x \wedge \bigvee_{e \in E} @_z \langle e \rangle x \rightarrow @_y z)$.

We get the following preliminary result:

Lemma 1 $Pre(\mathfrak{M})$ is a tree rooted at r if and only if $\mathfrak{M} \models_{hm} tree$ where $tree =_{def} \psi_1 \wedge \psi_2 \wedge \psi_3$.

Knowing that $Pre(\mathfrak{M})$ is a tree, in order to know if $\mathfrak{M}$ is an instance of a schema $\mathcal{G}$, we need now to express the conditions (ii) and (iii) of the definition of instance of a schema (Definition 15). These conditions are taken care by the following definitions :

DEFINITION 18 (PATTERN TRANSLATION). — The expression φ_{expr} is defined as :

1) If $expr$ is Λ then $\varphi_{expr} = \bigwedge_{e \in \mathcal{E}} \neg\langle e\rangle\top$

2) If $expr$ is a disjunctive pattern $expr_1 | \cdots | expr_j$ then $\varphi_{expr} = \bigvee_{i \in [1..j]} \varphi_{expr_i}$

3) If $expr$ is a conjunction pattern[5] $m_1, \cdots, m_j$ then:

let $\mathcal{E}_+ = \{e \mid (e\ N)^+$ appears in $expr\}$,
let $\mathcal{E}_! = \{e \mid (e\ N)!$ appears in $expr\}$,
let $\mathcal{E}_? = \{e \mid (e\ N)?$ appears in $expr\}$, and
let E_{expr} be the set of labels occurring in $expr$.

$$\begin{aligned}\varphi_{expr} = & \bigwedge_{e \in \mathcal{E}_+} \langle e\rangle\top \quad \wedge \\ & \downarrow x \bigwedge_{e \in \mathcal{E}_!} (\langle e\rangle\downarrow y\ (@_x[e]y)) \quad \wedge \\ & \bigwedge_{e \in \mathcal{E}_?} (\neg\langle e\rangle\top \ \vee\ (\downarrow x\ \langle e\rangle\downarrow y\ (@_x[e]y))) \quad \wedge \\ & \bigwedge_{e \in \mathcal{E} - E_{expr}} \neg\langle e\rangle\top\end{aligned}$$

In order to simplify the expression of the translation of a schema, we define the notation $\varphi_N =_{def} \varphi_{Expr(N)}$, where N is a non-terminal symbol (note that in the definition given below, $Symb(e)$ denotes a non-terminal symbol).

DEFINITION 19 (SCHEMA TRANSLATION). — Let $\mathcal{G}$ be a schema. Its translation $TRANS(\mathcal{G})$ is the formula

$$tree \wedge TRANS_1(\mathcal{G}) \wedge TRANS_2(\mathcal{G}) \qquad \text{where}$$

$TRANS_1(\mathcal{G}) =_{def} @_{root}\big(\varphi_{Root} \wedge \bigwedge_{e \in E} G^*[e]\varphi_{Symb(e)}\big)$, and
$TRANS_2(\mathcal{G}) =_{def} @_{root}\Big(\bigwedge_{\vec{e} \in \vec{E}} G^*[\vec{e}]\downarrow x \Big(\bigvee_{e \in Label(Symb(\vec{e})) \cap E} @_{root} F^*\langle e\rangle x\Big)\Big)$.

Intuitively, $TRANS_1(\mathcal{G})$ and $TRANS_2(\mathcal{G})$ respectively express conditions (ii) and (iii) of Definition 15.

Before giving the main result of this section, we illustrate the translation of a schema for semistructured data into a formula of hybrid multimodal logic for our running example and for the recursive schema of Example 13.

5. An elementary pattern is considered here as the particular case of a conjunctive pattern having just one element.

EXAMPLE 20 (CONTINUATION). — The formula associated to the grammar $\mathcal{G}_1$ defined in Example 9 is $TRANS(\mathcal{G}_1) = tree \wedge TRANS_1(\mathcal{G}) \wedge TRANS_2(\mathcal{G})$ where $TRANS_1(\mathcal{G})$ is

$$@_{root} \quad (\varphi_{Root} \wedge G^*[doc]\varphi_{Doc} \wedge G^*[publisher]\varphi_{Publisher} \wedge G^*[name]\varphi_{Name} \wedge G^*[article]\varphi_{Art} \wedge G^*[book]\varphi_{Book} \wedge G^*[auteur]\varphi_{Name} \wedge G^*[title]\varphi_{Name} \wedge G^*[date]\varphi_{Dat} \wedge G^*[isbn]\varphi_{Isb})$$

and
$TRANS_2(\mathcal{G})$ is

$$@_{root} \quad (G^*[\overrightarrow{cite}]\downarrow x \ (@_{root}F^*\langle doc\rangle x) \ \wedge \ G^*[\overrightarrow{publish}]\downarrow x \ (@_{root}F^*\langle book\rangle x) \wedge G^*[\overrightarrow{publishedby}]\downarrow x \ (@_{root}F^*\langle publisher\rangle x))$$

with

$$\varphi_{Root} =_{def} \bigwedge_{e\in\mathcal{E}-\{doc,publisher\}} \neg\langle e\rangle \top$$

$$\varphi_{Publisher} =_{def} \downarrow a \ \langle name\rangle\downarrow b \ (@_a[name]b) \wedge \bigwedge_{e\in\mathcal{E}-\{name,\overrightarrow{publish}\}} \neg\langle e\rangle \top$$

$$\varphi_{Doc} =_{def} \downarrow a \ \langle book\rangle\downarrow b \ (@_a[book]b) \wedge \downarrow a \ \langle article\rangle\downarrow b \ (@_a[article]b) \wedge \bigwedge_{e\in\mathcal{E}-\{book,article\}} \neg\langle e\rangle \top$$

$$\varphi_{Art} =_{def} \langle author\rangle \top \wedge \downarrow a \ \langle title\rangle\downarrow b \ (@_a[title]b) \wedge (\neg\langle date\rangle \top \vee \downarrow a \ \langle date\rangle\downarrow b \ (@_a[date]b)) \wedge \bigwedge_{e\in\mathcal{E}-\{author,title,date,\overrightarrow{cite}\}} \neg\langle e\rangle \top$$

$$\varphi_{Book} =_{def} \downarrow a \ \langle isbn\rangle\downarrow b \ (@_a[isbn]b) \wedge \bigwedge_{e\in\mathcal{E}-\{isbn,\overrightarrow{cite}\}} \neg\langle e\rangle \top \ \vee \ \langle author\rangle \top \wedge \downarrow a \ \langle date\rangle\downarrow b \ (@_a[date]b) \wedge \downarrow a \ \langle title\rangle\downarrow b \ (@_a[title]b) \wedge \downarrow a \ \langle \overrightarrow{publishedby}\rangle\downarrow b \ (@_a[\overrightarrow{publishedby}]b) \wedge \bigwedge_{e\in\mathcal{E}-\{author,date,title,\overrightarrow{cite},\overrightarrow{publishedby}\}} \neg\langle e\rangle \top$$

$$\varphi_{Name} =_{def} \bigwedge_{e\in\mathcal{E}} \neg\langle e\rangle \top$$

$$\varphi_{Dat} =_{def} \bigwedge_{e\in\mathcal{E}} \neg\langle e\rangle \top$$

$$\varphi_{Isb} =_{def} \bigwedge_{e\in\mathcal{E}} \neg\langle e\rangle \top$$

The model associated to the data graph given in Figure 1 is a model $TRANS(\mathcal{G}_1)$, hence such a graph is an instance of schema $\mathcal{G}_1$.

Example 21 is another example of schema translation, but in the case of a recursive schema.

EXAMPLE 21 (RECURSIVE SCHEMA TRANSLATION). — The formula associated to grammar $\mathcal{G}_3$ defined in Example 13 is $TRANS(\mathcal{G}_3) = tree \wedge TRANS_1(\mathcal{G}_3) \wedge TRANS_2(\mathcal{G}_3)$ where

$TRANS_1(\mathcal{G}_3)$ is $@_{root}(\varphi_{Root} \wedge G^*[lt]\varphi_{Tr} \wedge G^*[rt]\varphi_{Tr} \wedge G^*[leaf]\varphi_L)$, and $TRANS_2(\mathcal{G}_3)$ is $\top$

with

$$\varphi_{Root} \quad =_{def} \quad \langle tree\rangle\top \wedge \bigwedge_{e\in\mathcal{E}-\{tree\}} \neg\langle e\rangle\ \top$$

$$\varphi_{Tr} \quad =_{def} \quad \downarrow a\ \langle lt\rangle\downarrow b\ (@_a[lt]b) \ \wedge\ \downarrow a\ \langle rt\rangle\downarrow b\ (@_a[rt]b) \ \wedge\ \bigwedge_{e\in\mathcal{E}-\{lt,rt\}} \neg\langle e\rangle\ \top$$
$$\vee\ \downarrow a\ \langle leaf\rangle\downarrow b\ (@_a[leaf]b) \ \wedge\ \bigwedge_{e\in\mathcal{E}-\{leaf\}} \neg\langle e\rangle\ \top$$

$$\varphi_L \quad =_{def} \quad \bigwedge_{e\in\mathcal{E}} \neg\langle e\rangle\ \top$$

One can note that the translation is as simple as for the non recursive case.

We now present the main result of this section:

Theorem 3 Let $\mathcal{G}$ be a schema and let $TRANS(\mathcal{G})$ be its translation in hybrid multimodal logic. Then $\mathfrak{M}$ is an instance of $\mathcal{G}$ if and only if $\mathfrak{M} \models_{hm} TRANS(\mathcal{G})$.

Proof of Theorem 3 Before entering into the details of the proof, we first sketch its general structure. The proof is based on two main lemmas: Lemma 2 and Lemma 3.

In Lemma 2, we abstract from reference typing and show that:

Lemma 2 The data graph $\mathfrak{M}$ satisfies conditions (i) and (ii) of Definition 15 if and only if $\mathfrak{M} \models_{hm} tree \wedge TRANS_1(\mathcal{G})$.

Lemma 3 completes the previous one by taking into account reference typing:

Lemma 3 The following sentences are equivalent:

1) $\mathfrak{M} \models_{hm} tree \wedge TRANS_1(\mathcal{G})$ and $\mathfrak{M}$ satisfies condition (iii)
2) $\mathfrak{M} \models_{hm} tree \wedge TRANS_1(\mathcal{G}) \wedge TRANS_2(\mathcal{G})$.

The proof is rather complex and it needs some preliminary lemmas which are interrelated according to the dependency graph depicted below.

The following lemma, Lemma 4, says that when $Pre(\mathfrak{M})$ is acyclic, the cardinality of the operators $(+,*,!,?)$ occurring in a pattern is "well translated".

Lemma 4 Let $\mathfrak{M}$ be a rooted connected graph such that $Pre(\mathfrak{M})$ is acyclic. Let $L \subseteq \mathcal{E}$ be a set of labels, let *op* be in $\{+, *, !, ?\}$ and let s be a state of S. The two following properties are equivalent:

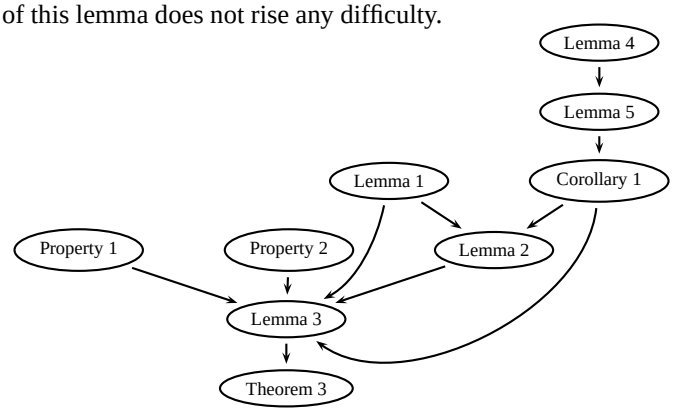
• for any $e \in L$, $(e\ N)^{op}$ matches $\mathfrak{M}$ at s,

• for any valuation g, we have $\mathfrak{M}, g, s \models_{hm} \psi_{op}(L)$ and for any $e \in L \cap E$, for any s_1 in $Out_e(s)$, $Expr(N)$ strictly matches $\mathfrak{M}$ at s_1, where

$\psi_*(L) =_{def} \top$, $\quad \psi_!(L) =_{def} (\downarrow x \bigwedge_{e \in L} (\langle e \rangle \downarrow y\ (@_x[e]y)))$,

$\psi_+(L) =_{def} \bigwedge_{e \in L} \langle e \rangle \top$, $\quad \psi_?(L) =_{def} \bigwedge_{e \in L} (\neg \langle e \rangle \top \vee (\downarrow x \langle e \rangle \downarrow y\ (@_x[e]y)))$.

The proof of this lemma does not rise any difficulty.



Before giving the other lemmas, we need a preliminary definition.

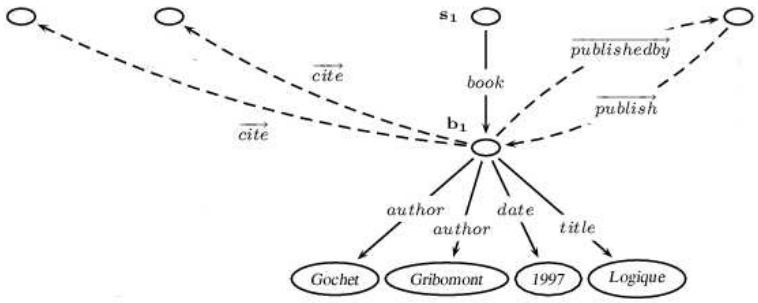
DEFINITION 22 (MAIN SUBGRAPH GENERATED BY s). — Let s be a state of S. The main sub-graph generated by s, noted $\mathfrak{M}_{/s}$, is defined as the sub-graph $(S_{/s}, s, R_{/s}, V_{/s})$ of $\mathfrak{M}$ rooted at s such that:

• $S_{/s}$ is the subset of S containing s and all the states which are reachable from s in $\mathfrak{M}$ by a non empty path whose sequence of labels $e_1, \cdots, e_{n-1}, e_n$ is such that, for any $i \in [1..n-1]$, $e_i \in E$ and $e_n \in \mathcal{E}$.

• $R_{/s}$ is the restriction of relations in R to the states of $S_{/s}$, and

• $V_{/s}$ is the restriction of V to the states of $S_{/s}$.

For instance, if $\mathfrak{M}$ is the model corresponding to the running example, Example 1, then $\mathfrak{M}_{/s_1}$ is the following graph:



Lemma 5 Let $\mathfrak{M}$ be a data graph such that $Pre(\mathfrak{M})$ is a tree. Let s be any state of S.

(A) Let $Expr(N)$ be Λ or a conjunctive pattern.

(A1) If s is a leaf in $Pre(\mathfrak{M})$, then the two following statements are equivalent:

– $Expr(N)$ strictly matches $\mathfrak{M}_{/s}$ at s,

– for any valuation g, we have $\mathfrak{M}_{/s}, g, s \models_{hm} \varphi_N$.

(A2) If s is not a leaf in $Pre(\mathfrak{M})$ and, for any state s_1 accessible from s by a non empty path in $Pre(\mathfrak{M})$ whose last edge is labelled by a $e \in E$, the following condition is satisfied :

(COND) the equivalence between the following properties holds:

– $Expr(M)$ strictly matches $\mathfrak{M}_{/s_1}$ at s_1

– $\mathfrak{M}_{/s_1}, s_1 \models \tau_M$ and, for any state s_2 accessible from s_1 in $\mathfrak{M}_{/s_1}$ by a path whose last edge is labelled by a $e \in E$, we have: $\mathfrak{M}_{/s_1}, g, s_2 \models_{hm} \varphi_{Symb(e)}$

then the two following properties are also equivalent:

– $Expr(N)$ strictly matches $\mathfrak{M}_{/s}$ at s,

– for any valuation g, we have $\mathfrak{M}_{/s}, g, s \models_{hm} \varphi_N$ and, for any state s_1 accessible from s by a path whose last edge is labelled by a $e \in E$, we have $\mathfrak{M}_{/s}, g, s_1 \models_{hm} \varphi_{Symb(e)}$.

(B) Let $Expr(N)$ be any pattern. The two following properties are equivalent:

– $Expr(N)$ strictly matches $\mathfrak{M}_{/s}$ at s,

– for any valuation g, $\mathfrak{M}_{/s}, g, s \models_{hm} \varphi_N$ and, for any state s_1 accessible from s by a path whose last edge is $e \in E$, we have $\mathfrak{M}_{/s}, g, s_1 \models_{hm} \varphi_{Symb(e)}$.

Sketch of proof : The results (A1) and (A2) are proved as consequences of Lemma 4.

The property (B) is a generalization of the above results covering also the disjunctive patterns. It is proved by induction on the size of the shortest path going from s to a leaf in $Pre(\mathfrak{M})$, noted $proff(s)$.

If $proff(s) = 0$ and $Expr(N)$ is Λ or a conjunctive pattern then the result is given by (A1). If $proff(s) = 0$ and $Expr(N)$ is a disjunctive pattern having the form $expr_1 | \cdots | expr_n$, where for each $i \in [1..n]$ $expr_i$ is a conjunctive pattern then by definition, $Expr(N)$ strictly matches $\mathfrak{M}_{/s}$ at s iff there is one $j \in [1..n]$ such that $expr_j$ strictly matches $\mathfrak{M}_{/s}$ at s. Hence, we can again use (A1) in order to conclude.

For the inductive step, for any state s_1 accessible from s by a non empty path in $Pre(\mathfrak{M})$ we have $proff(s_1) < proff(s)$, hence, by inductive hypothesis, we have the equivalence between the properties:

- $Expr(M)$ strictly matches $\mathfrak{M}_{/s_1}$ at s_1

• $\mathfrak{M}_{/s_1}, s_1 \models \tau_M$ and, for any state s_2 accessible from s_1 in $\mathfrak{M}_{/s_1}$ by a path whose last edge is labelled by a $e \in E$ we have: $\mathfrak{M}_{/s_1}, g, s_2 \models_{hm} \varphi_{Symb(e)}$

Hence, if $Expr(N)$ is Λ or a conjunctive pattern, then the condition (COND) is satisfied and (A2) is applicable, which leads to obtain the desired result. If $Expr(N)$ is a disjunctive pattern having the form $expr_1|\cdots|expr_n$, where for each $i \in [1..n]$ $expr_i$ is a conjunctive pattern then, by definition, $Expr(N)$ strictly matches $\mathfrak{M}_{/s}$ at s iff there is one $j \in [1..n]$ such that $expr_j$ strictly matches $\mathfrak{M}_{/s}$ at s. Hence, we can reduce this case to the previous one. □

We can then deduce the following corollary:

Corollary 1 Let $\mathfrak{M}$ be a data graph such that $Pre(\mathfrak{M})$ is a tree rooted at r. The two following properties are equivalent:

• $Expr(Root)$ strictly matches $\mathfrak{M}$ at r,

• for any valuation g, we have $\mathfrak{M}, g, r \models_{hm} \varphi_{Root}$ and, for any state s_1 accessible from r by a path whose last edge is $e \in E$, we have $\mathfrak{M}, g, s_1 \models_{hm} \varphi_{Symb(e)}$.

The proof of Lemma 2 comes easily: Condition (i) is handled thanks to Lemma 1 and Condition (ii) is handled thanks to Corollary 1.

Lemma 3 is proved using Lemmas 1 and 2, Corollary 1, and Properties 1 and 2 shown in Section 3.

The proof of Theorem 3 follows imediately from Lemma 2 and Lemma 3 .

Related and future work

We have already discussed the relation between our approach and [BUN 98] in Section 2.

In [ALE 01], a propositional dynamic logic with nominals, noted PDL^{PATH}, is proposed as a formalization of some classes of constraints on paths in semistructured data graphs. Such a logic is a variation of $Converse - PD$L with nominals [FIS 79] and has a decidable satisfiability problem. The authors show that such a logic allows one to formalize the so called "inclusion constraints", and they deduce that the implication problem for such constraints is decidable in exponential time.

• The issue of schema formalization is not addressed in that paper.

• The language of hybrid multimodal logic that we investigate in this paper allows us to express properties of graphs that PDL^{PATH} cannot express : this, thanks to the presence of state variables and of the binder $\downarrow$. However, [ARE 99] shows that the fragment of hybrid multimodal logic not containing the closure operators F and G, state variables and the binder $\downarrow$ is decidable, but that adding the binder $\downarrow$ leads to an undecidable logic.

• PDL^{PATH} allows one to express the transitive and reflexive closure of no matter which path expression, not only expressions denoting the union of accessibility relations as it is the case for our language.

An object of future work is the study of fragments of hybrid multimodal logic containing closure operators but either not containing the binder $\downarrow$ or else restricted forms of it.[6] This work might exploit the connection of such languages with the language of $Converse - PDL$ and it might lead to useful information on the decidability of fragments of hybrid multimodal logic, and, as a consequence, on the decidability of the implication problem for some classes of constraints.

A distinct but related topic is the expression of queries on semistructured data in hybrid multimodal logic. In this context, we plan, for instance, to investigate the issue of semantic query optimization and the issue of inclusion for XPATH queries, in the line of [DEU 01b, NEV 03].

As far as we know, our work is pioneer w.r.t. the definition of a schema for semistructured data powerful enough to allow typing of references and w.r.t. the constructive definition of a translation of schemas for semistructured data into formulas of a modal logic. The relation between our approach and [CAL 98] deserves further study.

4. References

[ABI 97a] ABITEBOUL S., "Querying Semi-Structured Data", *Proceedings of the 6th International Conference on Database Theory - ICDT'97, Delphi, Greece*, vol. 1186 of *Lecture Notes in Computer Science*, Springer, 1997, p. 1-18.

[ABI 97b] ABITEBOUL S., QUASS D., MCHUGH J., WIDOM J., WIENER J. L., "The Lorel query language for semistructured data", *International Journal on Digital Libraries*, vol. 1, num. 1, 1997, p. 68–88.

[ABI 97c] ABITEBOUL S., VIANU V., "Regular path queries with constraints", *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems - PODS'97, Tucson, Arizona*, ACM Press, 1997, p. 122–133.

[ABI 00] ABITEBOUL S., BUNEMMANN P., SUCIU D., *Data on the web: From Relations to Semistructured Data and XML*, Morgan Kauffman, 2000.

[ALE 01] ALECHINA N., DEMRI S., DE RIJKE M., "Path Constraints from a Modal Logic Point of View", *Proceedings of the 8th Int. Workshop on Knowledge Representation meets Databases - KRDB'01, Roma, Italy*, vol. 45, 2001.

[ARE 99] ARECES C., BLACKBURN P., MARX M., "A Road-map on Complexity for Hybrid Logics", *Proceedings of the 8th Annual Conference of the EACSL, Madrid*, LNCS, Springer, 1999, p. 307–321.

[ARE 01] ARECES C., BLACKBURN P., MARX M., "Hybrid Logics: Characterization, Interpolation and Complexity", *Journal of Symbolic Logic*, vol. 66, num. 3, 2001, p. 977–1010.

6. A fragment of hybrid multimodal logic not containing the closure operators F and G and containing only a restricted form of application of the binder $\downarrow x$ has already been proved decidable in [MAR 02].

[BID 03] BIDOIT N., CERRITO S., THION V., "Un premier pas vers la modélisation des données semi-structurées par la logique multi-modale hybride", report num. 1375, octobre 2003, L.R.I.

[BLA 95] BLACKBURN P., SELIGMAN J., "Hybrid Languages", *Journal of Logic, Language and Information*, vol. 4, 1995, p. 251–272.

[BLA 99] BLACKBURN P., TZAKOVA M., "Hybrid Languages and Temporal Logic", *Logic Journal of the IGPL*, vol. 7, num. 1, 1999, p. 27–54.

[BLA 00] BLACKBURN P., "Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto", *Logic Journal of the IGPL*, vol. 8, num. 3, 2000, p. 339–365.

[BUN 96] BUNEMAN P., DAVIDSON S., HILLEBRAND G., SUCIU D., "A query language and optimization techniques for unstructured data", *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, 1996, p. 505–516.

[BUN 97] BUNEMAN P., DAVIDSON S. B., FERNANDEZ M. F., SUCIU D., "Adding Structure to Unstructured Data", *Proceedings of the 6th International Conference on Database Theory - ICDT'97, Delphi, Greece*, vol. 1186, Springer, 1997, p. 336–350.

[BUN 98] BUNEMAN P., FAN W., WEINSTEIN S., "Path Constraints on Semistructured and Structured Data", *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems - PODS'98, Seattle, Washington*, ACM Press, 1998, p. 129–138.

[CAL 98] CALVANESE D., DE GIACOMO G., LENZERINI M., "Semi-structured Data with Constraints and Incomplete Information", *International Workshop on Description Logics - DL'98 , IRST, Povo - Trento, Italy*, vol. 11 of *CEUR Workshop Proceedings*, 1998.

[CAL 99] CALVANESE D., DE GIACOMO G., LENZERINI M., "Queries and Constraints on Semi-structured Data", *Proceedings of the 11th International Conference Advanced Information Systems Engineering - CAiSE'99, Heidelberg, Germany*, vol. 1626 of *Lecture Notes in Computer Science*, Springer, 1999, p. 434–438.

[Dal 03] DAL ZILIO S., LUGIEZ D., "XML Schema, Tree Logic and Sheaves Automata", *Proceedings of the 14th International Conference on Rewriting Techniques and Applications - RTA'03, Valencia, Spain*, vol. 2706 of *Lecture Notes in Computer Science*, Springer, 2003, p. 246–263.

[DCD] "Document Content Description", On `http://www.w3.org/TR/NOTE-dcd`, W3C Recommendation.

[DEU 01a] DEUTSCH A., TANNEN V., "Optimization Properties for Classes of Conjunctive Regular Path Queries", *8th International Workshop on Database Programming Languages - DBPL'01, Frascati, Italy*, vol. 2397 of *Lecture Notes in Computer Science*, Springer, 2001.

[DEU 01b] DEUTSCH A., TANNEN V., "Containment of Regular Path Expressions under Integrity Constraints", *Proceedings of the 8th International Workshop on Knowledge Representation meets Databases - KRDB'01, Rome, Italy*, vol. 45 of *CEUR Workshop Proceedings*, Technical University of Aachen (RWTH), 2001.

[FIS 79] FISHER N., LADNER R. E., "Propositional dynamic logic of regular programs", *Journal of Computer and System Science*, vol. 18, 1979, p. 194-211.

[FIT 83] FITTING M., *Proof Methods for Modal and Intuitionist Logic*, D.Reidel Publishing Company, 1983.

[FRA 03] FRANCESCHET M., DE RIJKE M., "Model Checking for Hybrid Logics", *Workshop Methods for Modalities*, 2003.

[HAC 95] HACID M.-S., RIGOTTI C., "Combining Resolution and Classifi cation for Semantic Query Optimization in DOOD", *Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases - DOOD'95, Singapore*, vol. 1013 of *Lecture Notes in Computer Science*, Springer, 1995, p. 447-466.

[HOS 00] HOSOYA H., VOUILLON J., PIERCE B. C., "Regular expression types for XML", *International Conference on Functional Programming - ICFP'00*, 2000, p. 11-22.

[HOS 03] HOSOYA H., PIERCE B. C., "Regular expression pattern matching for XML", *Journal of Functional Programming*, vol. 13, num. 6, 2003, p. 961–1004.

[KLA 00] KLARLUND N., MOLLER A., SCHWATZBACH M. I., "DSD: A Schema Language for XML", *3rd ACM Workshop on Formal Methods in Software Practice*, 2000.

[KRI 71] KRIPKE S., "Semantic Considerations on Modal Logic", LINSKI L., Ed., *Reference and Modality*, London, 1971, Oxford University Press, p. 63-72.

[MAR 02] MARX M., "Narcissists, Stepmothers and Spies", *International Workshop on Description Logics - DL'02*, 2002.

[MCH 99] MCHUGH J., WIDOM J., "Query Optimization for XML", *Proceedings of 25th International Conference on Very Large Data Bases - VLDB'99, Edinburgh, Scotland, UK*, Morgan Kaufmann, 1999, p. 315–326.

[NEV 03] NEVEN F., SCHWENTICK T., "XPath containment in the presence of disjunction, DTDs, and variables", *Proceedings of the 9th International Conference on Database Theory - ICDT'03, Siena, Italy*, vol. 2572 of *Lecture Notes in Computer Science*, Springer, 2003.

[RAY 01] RAY E. T., *Learning XML*, 0'Reilly Associates, 2001.

[WAL 90] WALLEN L. A., *Automated Deduction in Nonclassical Logics: Efficient Matrix Proof Methods for Modal and Intuitionistic Logics*, MIT Press, 1990.

[XML] "XML Schema", On `http://www.w3.org/TR/xmlschema-0/`, W3C Recommendation.