# PROOF SYSTEMS FOR DATA QUERIES

An ANR "Young Researcher" Proposal

## Contents

## Summary

Semi-structured data, in particular in the form of XML documents, is now the established paradigm for storing and retrieving data over the Internet. XPath is a querying language, which allows to select elements in XML documents. Its use is pervasive in Web-oriented languages like XSLT or XQuery, but also in general-purpose languages like Java or C#. It combines the ability to navigate the XML document—which finds its roots in modal logic—with that of comparing data values found in several, distantly related elements. The static analysis of data queries, expressed in XPath or in related data logics, aims to optimize and verify queries.

The presence of data tests makes the formal verification of XPath queries undecidable. The research on restrictions and variants of XPath must therefore seek new standards, algorithms, and languages, which should strike a balance between practical usability—can typical queries be expressed in the restricted language?—and amenability to formal analysis—can we check queries written in the language? Our main objective is to provide rigorous foundations for automatic analysis tools that verify queries in data-centric programs.

The project explores a new avenue at the interface between several research communities in database theory, infinite-state system verification and proof theory: the main thrust behind the project is

the investigation of proof-theoretic tools for data logics, using in particular insights from substructural logics. Along the way, it touches on several difficult open problems in these communities.

In more details, the first task aims at developing proof systems for data logics. We hope to draw inspiration notably from proof-theoretic approaches to modal logics and automata. The connection between data and substructural connectives as such relies on the second task, which aims in particular at using structures with data as models of substructural logics. The third task should further the use of counter systems as a means to obtain algorithms and complexity results on both data and substructural logics. Although the project has a theoretical streak, we wish to integrate practical insights: the first task should lead to the development of a prototype checker for XPath, and we also plan to construct a benchmark suite for XPath with data.

KEY WORDS. Database, verification, XML, queries, proof systems, substructural logic

## CONSORTIUM DESCRIPTION

| Partner | Last Name | First Name | Current Position | Implication (pers.month) | Responsibilities |
|---|---|---|---|---|---|
| LSV | Baelde | David | MdC | 19.2 (40%) | Task A |
| LSV | Schmitz | Sylvain | MdC | 38.4 (80%) | Task 0, Task B, Task C |

**Partner**. The *Laboratoire Spécification et Vérification* (LSV) is a shared research unit between ENS Cachan, CNRS, and INRIA, located on the campus of ENS Cachan. The lab, founded in 1997, is a leading expert in modeling and automatically verifying software systems. Inside the lab, the *Dahu* axis is an INRIA project headed by Luc Segoufin, which specializes on the specification and verification of data-centric systems, in particular distributed database systems. The group has an especially strong record on satisfiability problems for data queries, having co-authored for instance [15, 27, 65, 16, 66, 31, 28].

**Participants**. The consortium is based at LSV, and includes two permanent participants (Sylvain Schmitz and David Baelde) and a Ph.D. student that will be funded by the project. We ask for a three-years Ph.D. fellowship along with funding for research visits. The total duration of the project is of four years, which makes it easier to find and integrate a Ph.D. student.

*Sylvain Schmitz* (principal investigator) is an associate professor (*maître de conférences*) at ENS Cachan. His main area of research is the algorithmic complexity of logical theories and formalisms, with in particular an expertise on the complexity results derivable from the use of *well quasi orderings* (see e.g. Figueira et al., 2011). He has also worked on the complexity of verifying concurrent systems with data (Haddad et al., 2012), on the complexity of satisfiability in the navigational fragment of XPath (Boral and Schmitz, 2013), and on the complexity of provability in substructural logics (e.g. Lazić and Schmitz, 2014).

After five years in the Infini axis at LSV on the verification of infinite-state systems (where he participates since 2011 in the ANR Blanc REACHARD project on the algorithmics of reachability problems in counter systems), he moved to Dahu in September 2013 where he benefits from an INRIA-funded sabbatical. This proposal of using proof systems for XPath satisfiability is an entirely novel direction within Dahu.

Selected relevant publications:

D. Figueira, S. Figueira, S. Schmitz, and Ph. Schnoebelen. Ackermannian and primitive-recursive bounds with Dickson's Lemma. In *LICS 2011*, pages 269–278. IEEE, 2011. doi:10.1109/LICS.2011.39.

S. Haddad, S. Schmitz, and Ph. Schnoebelen. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *LICS 2012*, pages 355–364. IEEE, 2012. doi:10.1109/LICS.2012.46.

A. Boral and S. Schmitz. Model checking parse trees. In *LICS 2013*, pages 153–162. IEEE, 2013. doi:10.1109/LICS.2013.21.

R. Lazić and S. Schmitz. Non-elementary complexities for branching VASS, MELL, and extensions. In *CSL-LICS 2014*. ACM, 2014. To appear, arXiv:1401.6785 [cs.LO].

*David Baelde* is an associate professor (*maître de conférences*) at ENS Cachan. He has a strong background in proof theory and proof search, and has worked on fixed points in linear logic (Baelde, 2012) as well as nominal quantification in sequent calculi (Baelde, 2009a). Since his arrival at ENS Cachan in September 2012, he has accentuated his focus on bridging the gap between proof theory and verification: he is investigating connections between automata theory and cyclic and infinite proofs (e.g. Baelde, 2009b) and working on the verification of cryptographic protocols as a member of the SECSI axis at LSV.

Selected relevant publications:

D. Baelde. On the expressivity of minimal generic quantification. In *LFMTP 2008*, volume 228 of *Elec. Notes in Theor. Comput. Sci.*, pages 3–19. Elsevier, 2009a. doi:10.1016/j.entcs.2008.12.113.

D. Baelde. On the proof theory of regular fixed points. In *TABLEAUX 2009*, volume 5607 of *Lect. Notes in Comput. Sci.*, pages 93–107. Springer, 2009b. doi:10.1007/978-3-642-02716-1_8.

D. Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Logic*, 13(1), 2012. doi:10.1145/2071368.2071370.

*External Collaborators.* The project also requests funding for research visits to and from several international collaborators, in particular *Diego Figueira* (U. Edinburgh, Scotland), *Santiago Figueira* (U. Buenos Aires, Argentina), *Sławomir Lasota* (U. Warsaw, Poland), *Ranko Lazić* (U. Warwick, England), and *Nikos Tzevelekos* (U. College London, England).

## Changes Since the Pre-Proposal

There are no important changes in this detailed proposal compared to the pre-proposal. Two long-standing open problems mentioned as interesting preliminaries in Section 1.2.4 of the pre-proposal, namely determining the complexities of provability in affine linear logic and in implicational relevance logic, have been solved in the meantime respectively by Lazić and Schmitz [51] and Schmitz [62].

## 1. Context, Positioning & Objectives

The advent of the Internet as a universal medium for communication deeply modifies the way software is developed. Today's programs are run and sold over the network; they rely on distributed data management systems, and the added value of many Web applications resides in the original way they extract and process data from several sources, i.e. in their *data queries.* The standards of the Web, in particular XML or XPath, are now part of the standard toolkit of software developers.

As we increasingly rely on Web applications for many aspects of our lives, from e-government to health self-monitoring applications and social networks, the safety requirements of such data-centric programs become more and more difficult to ignore. Moreover, the data querying languages have proven to be extremely challenging for verification, spanning the investigation of diverse variants and extensions, but still lacking unifying theories. The purpose of this project is to deepen the formal foundations to data-centric reasoning by considering its relationships with substructural proof systems.

### 1.1. **The Verification of Data Queries.**

1.1.1. *XML as a Database.* The relational data model, introduced by Codd more than 40 years ago and still in use in countless applications, presents data as tables (see Figure 1 for an example). Each row of a table then represents a tuple in a relation.

***Students***

| *id* | *name* |
|---|---|
| s1 | John |
| s2 | Mary |
| s3 | Steve |

***Enrollment***

| *sid* | *cid* | *grade* |
|---|---|---|
| s1 | c1 | 10 |
| s1 | c2 | 12 |
| s3 | c1 | 17 |

***Classes***

| *id* | *name* | *instructor* |
|---|---|---|
| c1 | Programming | John Doe |
| c2 | Algebra | Jane Roe |

Figure 1. Example of a relational database.

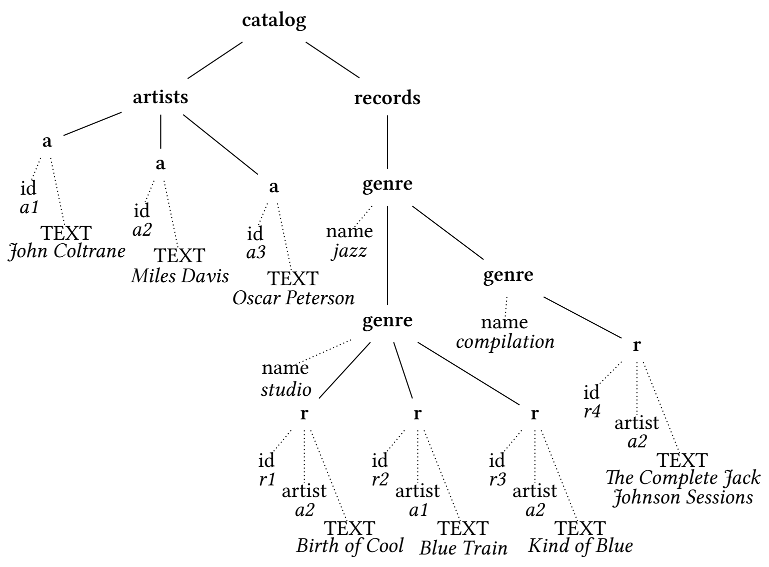
By contrast, the standard format for data exchange over the Internet is XML (standing for *eXtended Markup Language*), a representation for hierarchical data. While an XML document is a simple human-readable text file (see Figure 2a for an example), it really represents a tree with data values decorating the nodes (see Figure 2b for the corresponding tree[1]). Compared to classical relational databases, the tree topology provides a hierarchical structure to information. While early theoretical works on XML ignored the difficulties of data values and concentrated on the tree structure, the focus has now shifted to *data trees* and to the values found in the nodes.

Although our examples in this proposal will focus on XML documents, this is not the only format for hierarchical data on the Internet: for instance, JSON is another popular, human-writable alternative based on attribute-value structures, and JSON data can also be interpreted in the more abstract setting of data trees.

[1]Technically, such a data tree is a finite, unranked, ordered tree with node labels drawn from $E \times \mathbb{D}^A$ where $E$ and $A$ are finite alphabets of element and attribute names respectively, and $\mathbb{D}$ is an infinite data domain. The attribute-to-datavalue mapping of a node is presented using children connected through dotted lines in Figure 2b; note that text contents are seen as attributes.

```
<catalog>
  < artists >
    <a id="a1">John Coltrane</a>
    <a id="a2">Miles Davis</a>
    <a id="a3">Oscar Peterson</a>
  </ artists >
  <records>
    <genre name="jazz">
      <genre name="studio">
        <r id="r1" artist ="a2">Birth of Cool</r>
        <r id="r2" artist ="a1">Blue Train</r>
        <r id="r3" artist ="a2">Kind of Blue</r>
      </genre>
      <genre name="compilation">
        <r id="r4" artist ="a2">The Complete Jack
          Johnson Sessions </r>
      </genre>
    </genre>
  </records>
</catalog>
```

(A) Example of an XML document.



(B) Corresponding DOM data tree.

FIGURE 2. XML as a hierarchical data representation.

1.1.2. *Data Queries.* XPath is a querying language, which allows to *select* elements in XML documents documents. Its use is pervasive in Web-oriented language like XSLT or XQuery, but also in general-purpose languages like Java or C#. It combines the ability to navigate the XML tree—which finds its roots in modal logic—with that of comparing data values found in several, distantly related nodes [11].

Alongside with the *evaluation* of the set of elements selected in a document, the main computational problem associated with a query is the *satisfiability* problem: given an XPath query, and optionally a schema for the type of XML documents we want to consider, does there exist at least one XML document on which this query would select some node? This abstract question actually allows to answer a large number of questions on the reliability and performance of a query; to wit:

**usefulness**: is the data query useful at all, i.e. will it select some parts of a well-formed document? The question is far from trivial in an environment where the data can be retrieved from external sources, prone to changes in their structure. This allows simple optimizations in batch processes, with significant savings [41].

**optimization**: can a query be replaced by another, simpler query? This is a classical question in query rewriting in database theory. Although naive evaluation algorithms are hopefully not in use anymore [11], the size of a query still has a significant practical impact on the complexity of enumerating its answers. This type of optimization has been applied to XPath queries without considering data values [36], with an appreciable impact on performance.

**confidentiality**: can a query return privileged data in a document *access control* scenario? A database management system might offer a general querying engine but want to restrict access to confidential information. The most basic check should be that this information cannot appear in any answer. (More involved checks involve verifying that privileged data can only appear in aggregate form, for instance to prevent privacy issues with medical data.)

As an illustration of the data-manipulating primitives found in XPath, Figure 3 presents two XSLT snippets of a toy program producing HTML documents from XML-formatted catalogs like the one in

```
<xsl:template match="artists">
  <ul>
    <xsl:apply-templates select="a[@id=//r/@artist]"/>
  </ul>
</xsl:template>
```

(A) Filtering artists to only include those with records in the catalog.

```
<xsl:template match="r">
  <span class="artist">
    <xsl:apply-templates select="id(@artist)"/>
  </span>
  <xsl:text>: </xsl:text>
  <span class="title">
    <xsl:value-of select="."/>
  </span>
</xsl:template>
```

(B) Recovering the artist for the current record.

FIGURE 3. Examples of XPath queries (framed) embedded inside XSL templates.

Figure 2a. XPath queries (in magenta) are employed there to relate distant XML elements using data values. Those particular examples of queries exercise two of the most involved primitives available in XPath:

- The query in Figure 3a matches the attribute values in two different elements, i.e. performs a data *join*. Its evaluation on the '**artists**' element of the XML document of Figure 2a would disregard the element '**a**' with content '*Oscar Peterson*.'

  From a database perspective, joins are the single most important construct in a query language. However, the current concrete algorithms for the static analysis of XPath queries are limited to fragments without data tests.
- The query in Figure 3b jumps to the element with a specific identifier, i.e. navigates the XML document based on the implicit connections between identifiers and their references. The evaluation of this query on the document of Figure 2a would find e.g. that '*Miles Davis*' is the associated artist name for the element '**r**' corresponding to the '*Birth of Cool*' record.

  Identifier jumps are an important construct in practical uses of XPath. They are similar to the '@' operator in hybrid modal logic [4], and rely on key and foreign key constraints in the XML schema. They are systematically ignored in the literature on static XPath analysis, whether theoretical or practical.

These are not the only difficulties when tackling the verification of XPath queries. For instance, many real-life XPath queries perform calls to arithmetic and string-manipulating functions, which lead to an undecidable satisfiability on their own. Also, the literature on fragments of XPath with data only considers a simplified syntax based on version 1.0 of XPath defined in 1999. In the meantime, XPath 2.0 became an official W3C recommendation in 2010, with added complexity due to path intersections, `for` loops, etc. [see 65, for a study of XPath 2.0 *without* data]. Version 3.0 was adopted very recently in April 2014[2] and includes for instance anonymous function definitions. Thus XPath includes more and more general programming constructs, and is hardly a domain-specific language for path navigation any longer. For verification purposes, we focus nevertheless in this proposal on the two previous types of data-manipulating primitives along with XPath 1.0-style navigation, which already provide the core features.

1.1.3. *Operational Models & Algorithmics.* The presence of data tests makes the satisfiability problem for general XPath queries algorithmically impossible. The theoretical research on *data logics* has therefore focused on decidable restrictions and variants [15, 27, 44, 31]. The decidability frontier

[2]See the recommendation at http://www.w3.org/TR/xpath-30/.

depending on the allowed primitives is now getting clearer, and various complexity vs. expressiveness trade-offs have been investigated, with satisfiability problems requiring from exponential to Ackermannian computing resources.

However, positive results for data-aware static analyses of XPath queries are still rather scarce, arguably because they are still difficult to establish and to understand. This severely hinders the search for practically useful fragments of XPath amenable to formal verification. These positive results are most often grounded in the identification of a suitable model of *data tree automaton*, which compiles out the navigational part. Two main difficulties then arise on the choice of a suitable data automaton model:

(1) It has to be expressive enough to handle the data logic—here typically model-theoretic arguments are employed to prove the reduction from satisfiability to the emptiness problem for data automata.
(2) It has to remain decidable—several results rely on a reduction from the emptiness problem for data automata to a reachability problem on *counter systems*, where the numbers of distinct data values are modeled through non-negative integers. Crucially, these counter values are usually not *reliable*, as some data values initially deemed different might actually be the same without impairing the existence of a model. Decidability and complexity results then stem from results on counter systems, prominently the theory of well quasi orderings [63].

Due to these strong constraints, each fragment or variant of XPath has to rely on a *different* definition of a data automaton, tailored to its exact navigational power and data test capabilities. A consequence of these difficulties is that the exact complexity in the most promising decidable fragments [e.g. 32] is currently unknown, which is a sign that their algorithmic properties are still eluding us. It seems to us that the field is still very young and in need of better abstractions.[3]

1.2. **Substructural Logics** like *linear logic* [38] are resource-conscious approaches to logic: whereas classical logic sees propositions as permanent Aristotelian truths, linear logic sees them as consumable resources, and allows for a fine-grained control over the structural rules of *weakening* and *contraction*. Substructural logics include for instance linear, relevance, and separation logics.

In the context of data logics, the resources we want to capture with substructural logics are distinct *data values*. Strikingly, although the connection between substructural logics and counter machines strongly suggests it, the substructural viewpoint has *not* been exploited in connection with data logics. We believe this is an important untapped source for a unifying theory of data queries.

1.2.1. *Reasoning with Resources.* In order to better illustrate how substructural logics can be applied to reason about resources, let us consider a semantics based on finite sets $s$ of resources from some domain $\mathbb{D}$. Such sets are equipped with a commutative and associative partial action '$\uplus$' for disjoint unions. We can then define a satisfiability relation for a logic with a product '$\bullet$' inspired by the logic of *bunched implications* (BI) of O'Hearn and Pym [58]:[4]

$$s \models A_1 \bullet A_2 \qquad\qquad \text{iff } \exists s_1, s_2 \text{ s.t. } s \supseteq s_1 \uplus s_2, s_1 \models A_1, \text{ and } s_2 \models A_2 \;.$$

[3]The work that comes closest to our wishes is the definition of *class automata* by Bojańczyk and Lasota [14], which unifies several data automata formalisms. It is however unclear how to leverage this framework to obtain fine algorithmic characterizations.

[4]We make no claims of completeness whatsoever: our toy logic actually has little in common with BI.

Intuitionistic Boolean connectives and atomic propositions for elements '$d$' in $\mathbb{D}$ can be added to our logic by:

$$\begin{aligned}
&s \models d && \text{iff } d \in s \ , \\
&s \models A \wedge B && \text{iff } s \models A \text{ and } s \models B \ , \\
&s \models A \vee B && \text{iff } s \models A \text{ or } s \models B \ , \\
&s \models A \to B && \text{iff } \forall s' \supseteq s, s' \models A \text{ implies } s' \models B \ , \\
&s \models \bot && \quad \text{never} \ , \\
&s \models \top && \quad \text{always} \ .
\end{aligned}$$

We can observe some interesting behaviors for our products compared to e.g. Boolean conjunction '$\wedge$': for instance, $s \models A$ does not necessarily imply $s \models A \bullet A$, e.g. on the one hand $\{d\} \not\models d \bullet d$ and on the other hand $\{d, d'\} \models \top$ and $\{d, d'\} \models \top \bullet \top$. On the other hand, $s \models A \bullet B$ implies $s \models A$ and $s \models B$, and thus $s \models A \wedge B$ (using monotonicity: $s \subseteq s'$ and $s \models A$ together imply $s' \models A$). Thus our product operator can track resources with some precision, and interacts with additive connectives like '$\wedge$' or '$\top$.' This type of distinctions is at the heart of resource-based semantics for substructural logics [67, 38, 58, 22].

Substructural logics have been considered to reason about finite trees (see the spatial logic of Calcagno et al. [21]). With our toy example, the set of resources $s$ could be seen as the set of data values found in a node and its descendants in a binary data tree; the sets $s_1$ and $s_2$ then being the corresponding sets for its two children—the disjointness of $s_1$ and $s_2$ then meaning that we work on *disjoint data trees*, where no data values can be shared between incomparable nodes in the tree topology. This is only meant as an illustration of why we think substructural logics seem rather fit to reason about data trees as well.

1.2.2. *Proof Theory.* Substructural logics come with a rich theory of syntactic *proof systems*, allowing to establish the truth of statements by proof search. For instance, a multiplicative connective like '$\bullet$' above would typically spawn the following two rules in a Gentzen-style sequent calculus:

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \bullet B \vdash C}\,(\bullet_{\mathsf{L}}) \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \bullet B}\,(\bullet_{\mathsf{R}})$$

where $A, B, C$ denote formulæ and $\Gamma, \Delta$ denote multisets of formulæ. The word *substructural* itself refers to the absence of some substructural rules, e.g. we saw before that the following *contraction* rule (C) would not be sound for our logic, but the *weakening* rule (W) would be:

$$\frac{\Gamma, A, A \vdash C}{\Gamma, A \vdash C}\,(\mathsf{C}) \qquad \frac{\Gamma, A \vdash C}{\Gamma, A, B \vdash C}\,(\mathsf{W})$$

The study of linear logic in sequent calculus has brought a deeper understanding to an already well trodden field. An excellent example of an outcome of investigations in linear logic is the notion of *focusing*: it brings proof systems where the shape of proofs is significantly more constrained, thus enabling efficient proof search algorithms. While focusing has originally been designed for linear logic by Andreoli [3], its scope is much wider than that: focused proof systems have since been devised for various logics, linear and non-linear [7, 53, 10]—although all results ultimately rely on a linear decomposition of logical connectives. Besides proof search efficiency concerns, focusing also significantly eases the encoding of computations into proofs [9, 57] and may thus be used to prove complexity lower bounds on provability [62].

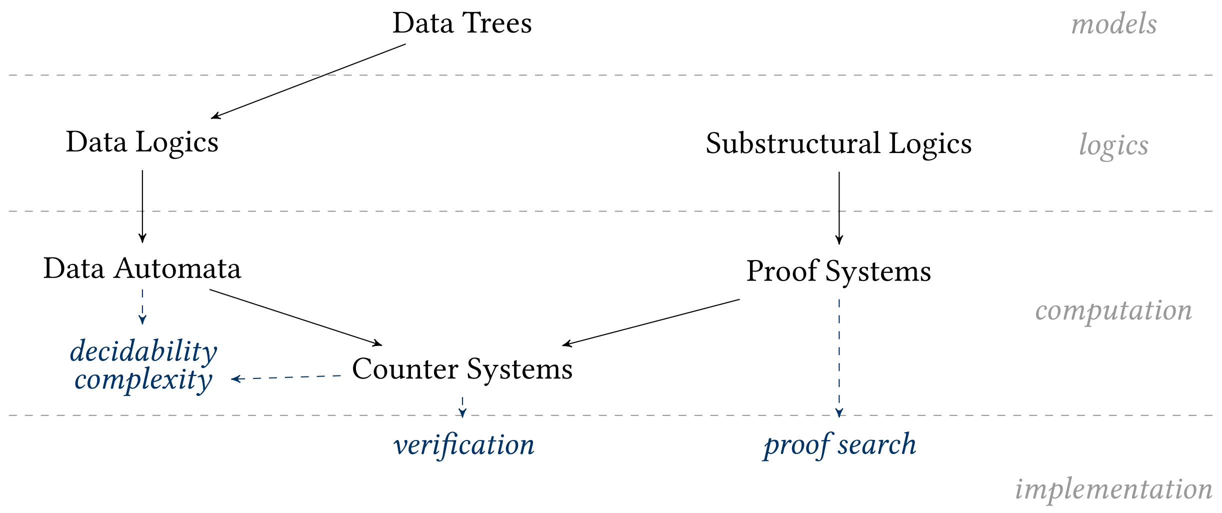


FIGURE 4. Summing up the context of the proposal.

Unfortunately, not all substructural logics enjoy such a clear (and helpful) expression in a sequent calculus as linear logic does. For separation logic, coming up with suitable sequent (or display or tableaux or ...) calculi is considered as a challenging issue [see e.g. 19, 43, for some results]. Similarly, sequent systems for involved modal logics like *propositional dynamic logic*—which is the main inspiration for the navigational fragment of XPath [2]—often rely on complex notations based on deep (aka nested) sequents or on semantic labels. A rather elegant solution is then to make such semantic labels part of the syntax and to develop a theory of *nominals* for them, i.e. of atomic propositions holding in only one single world. This is the case in *hybrid modal logic* [4]; closer to our concerns with data logics, see [5] for an application to *memory logics.*

1.2.3. *Computational Complexity.* The full propositional linear logic is undecidable [54], and so are relevance logic [68] and propositional separation logic [49, 20]. Similarly to satisfiability in data logics, provability in several fragments has been shown decidable using counter systems and well quasi orderings [45, 69, 46, 25].

The key observation to this end, when considering rules like ($\bullet_\mathsf{L}$) and ($\bullet_\mathsf{R}$) above in a cut-free sequent calculus, is that they satisfy the *subformula property*: the formulæ appearing in the rule premises are subformulæ of those in the conclusion. This means that, when attempting to decide whether a formula $A$ is valid, it suffices to look for proofs of $\vdash A$ that only employ formulæ from $\mathrm{Sub}(A)$ the set of subformulæ of $A$. Counter systems appear naturally in this context when we see a sequent $\Gamma \vdash B$ as a configuration $(q_B, \vec{v}_\Gamma)$ in $Q \times \mathbb{N}^d$ of a counter system, associating a state $q_B$ in $Q = \mathrm{Sub}(A)$ with a counter valuation $\vec{v}_\Gamma$ of dimension $d = |\mathrm{Sub}(A)|$. Updating counter values and states according to the rules of the sequent calculus is then routine work; for instance, the rules ($\bullet_\mathsf{L}$) and ($\bullet_\mathsf{R}$) can typically be implemented in a *branching vector addition system with states* (BVASS, see [25, 51, 62]). Adding the weakening rule (W) means that we will deal with *lossy* BVASS, for which reachability has recently been proven TOWER-complete [51].

Provability in a sequent system thus translates into a reachability problem in a counter system. This reduction to reachability problems allows to apply techniques inherited from the rich literature on the verification of counter systems: e.g. decidability using well-quasi-orders [33, 63], complexity bounds using combinatorial analyses and encodings of machines on a budget [55, 59, 64, 50, 12, 29, 26, 52, 51, 62, 24].

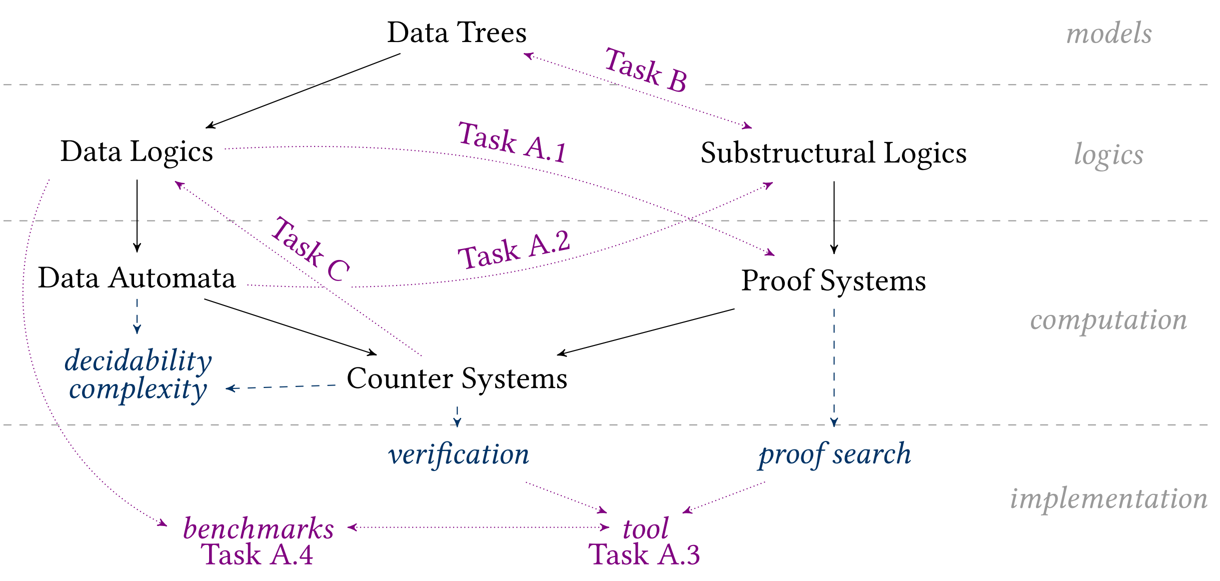


FIGURE 5. The objectives of the proposal: "connecting the dots."

The most challenging problem in this area is whether reachability is decidable in the aforementioned BVASS model: de Groote et al. [25] introduced the model and showed that BVASS reachability is recursively equivalent to provability in MELL the *multiplicative exponential* fragment of linear logic, a problem open since the early '90s. Although decidability is unknown, complexity lower bounds have been derived first by Lazić [50, 2EXPSPACE-hardness], and recently improved by Lazić and Schmitz [51, TOWER-hardness]. The importance of the problem is better appreciated when one knows that BVASS have actually been independently defined in several areas, including computational linguistics [60, 61] and protocol verification [70], and are furthermore related to the satisfiability of data logics [15, 30] and the verification of recursive parallel programs [18].

1.3. **Objectives.** The picture we just drew of the issues surrounding XPath satisfiability comprises two main components (see Figure 4):

(1) On the left and as developed in Section 1.1, XML documents, seen as data trees, can be queried using data logics like XPath. Formal reasoning on such queries most often relies on defining a suitable model of data automaton, which then provides decidability and/or complexity results, often using the theory of counter systems as an extra step.
(2) On the right and as developed in Section 1.2, substructural logics are endowed with resource-conscious proof systems, for which again algorithmic results often go through the theory of counter systems.

The common ground here are counter systems: importantly, essentially the *same* classes of counter systems are employed in both cases. This proposal aims at lifting this observation and "connecting the dots": we intend to build direct bridges between data logics and substructural logics and to examine their consequences (see Figure 5). We describe the objectives of the proposal in the following.

1.3.1. *Proof Systems for Data Queries.* An issue with the current results on the satisfiability of XPath fragments and other data logics is that each variant has its own decidability proof, leading to a fragmented landscape of results (cf. §1.1.3). The ultimate objective here is to *unify* such results, using a syntactic approach, by developing proof systems for XPath, for instance in the form of sequent calculi.

**Task A**: *Design and implement proof system(s) for data logics.*

The hope is that, depending on the navigational primitives, our proof systems would enjoy good properties that would lead to decidability and efficient procedures. For instance, the subformula property and a monotonicity property would together allow to apply the theories of well quasi orderings and of counter systems (cf. §1.2.3).

We expect this objective to be difficult to attain. Nevertheless, an encouraging fact is that the automata-based approaches often come close to a presentation as deduction systems (see Task A.2). A second encouraging fact is that, although we are not aware of any proof systems for XPath with data, the case *without* data has been thoroughly studied [e.g. in 66]. These provide ready-to-use solutions to describe the navigational aspects of XPath in a proof system. A promising angle for their extension to data logics is to work with *hybrid* logics (cf. §1.2.2); see Task A.1.

1.3.2. *Data Models for Substructural Logics.* As hinted in §1.2.1, we suspect that substructural logics could be used fruitfully to reason on data trees. The main objective here is to clarify this view of structures with data, like XML documents, as models for substructural logics. We will study which logical connectives to use and how they should interact with the classical Boolean connectives—many such combinations have already been investigated in the context of linear, relevance, and separation logics. The influence of key and foreign key constraints in XML documents should also be considered here. The results of these investigations will feed the other two tasks.

**Task B**: *Investigate the model-theoretic relations between data and substructural logics.*

1.3.3. *Data Logics and Counter Systems.* We intend to further study the relations between data logics and counter systems. The connections between data logics and counter systems are rather well-understood in the case of *data words*, i.e. of data trees of degree one, see e.g. [17, 28]. The picture is less clear in the case of general data trees: for instance Bojańczyk et al. [15] show how to encode BVASS executions in a data logic on trees, but regarding the converse direction Dimino et al. [30] needed to extend the BVASS model in order to exhibit an encoding of the same data logic into counter systems. This begs the question whether such an extension is really necessary.

More generally, we wish to investigate the landscape of data logic fragments and BVASS variants:

**Task C**: *Study the relations between branching counter systems and data logics on trees.*

One of the motivations is that, if we managed to provide a data logic whose satisfiability problem is recursively equivalent to BVASS reachability, we would then have a different angle for attacking the open problem of deciding BVASS reachability.

1.3.4. *Transverse Consideration: Freshness.* One of the crucial points in the current decidability arguments is that data logics can only enforce a limited *freshness* of data values. However, real-world XML schemas come with integrity constraints (such as keys and foreign keys), and full XPath comes with "`id`" modalities for identifier jumps (cf. Figure 3b), which explicitly concern unique data values. We want to investigate how this impacts the theory of data values. In particular, we would like to determine the decidability frontier for data logics with uniqueness constraints. As we witnessed a large number of occurrences of identity jumps in actual XPath queries (from the DocBook XML distribution), we suspect that this would greatly improve the practical usability of a satisfiability checker (see §1.3.6). This concern should permeate all our tasks.

1.3.5. *Transverse Consideration: Complexity.* One of our objectives is to better understand the source of high complexities in deciding data logics, using combinatorial techniques from well quasi orderings and counter machines to derive direct bounds on proof systems (cf. §1.2.3). We see computational complexity results as excellent guides for algorithmic considerations.

We have recently made two breakthroughs [51, 62] on long-standing open problems on the complexity of propositional substructural logics: implicational relevant logic [47] (shown decidable in 1959) and affine linear logic [46] (shown decidable in 1995), which allow respectively unrestricted structural contraction and weakening in linear logic. We wish to pursue this line of inquiry whenever relevant to the project.

1.3.6. *Concrete Algorithms* for XPath satisfiability are currently limited to fragments that include neither data tests nor identifier jumps [e.g. 37]. They are quite efficient on realistic small inputs in spite of high worst-case complexities, and can help discover substantial query optimizations [36, 41]. One of our objectives in this project is the development of a prototype satisfiability checker for XPath with data, see Task A.3.

A related issue we found while preparing this project proposal is the lack of a suitable benchmark suite for XPath with data. The few benchmark suites we found target the evaluation of XPath queries on specific inputs, rather than their static analysis independently of any given input. As a result, their coverage of data primitives varies from nonexistent to wholly insufficient; for instance, XPathMark [34] contains a single instance of a query with an identifier jump, and no data joins at all. This entails that the practical relevance of the various fragments of XPath studied in the literature has not really been investigated. For instance, a cursory inspection of XPath queries in a large XSLT development (DocBook XML) shows that identifier jumps are employed much more often than data joins. Other uses of XPath, e.g. in XQuery scripts, might yield different results [1]. One of our objectives in this project is therefore the compilation of a benchmark suite for XPath with data, see Task A.4.

1.4. **Positioning.** The project's themes are at the conjunction of different communities in database theory, proof theory, and verification, related for instance to two special interest groups of the association for computing machinery (ACM): SIGMOD on the management of data, and the newly chartered SIGLOG on logic in computer science.

The sole partner of the project is the *Laboratoire Spécification et Vérification* (LSV) at ENS Cachan, which has a leading position in verification. The research will take place in the INRIA Dahu project led by Luc Segoufin, which has an internationally recognized expertise on formal methods for data-aware verification.

The proposal is related to several recent projects:

**FCT FoX (2009–2012):** on *Foundations of XML* (http://fox7.eu/) was a networking programme funded by the European Research Consortium and coordinated by Luc Segoufin at LSV. Our proposal is in the direct line of FoX, which also investigated formal methods for XML processing. Many of the recent advances on XPath satisfiability with data were initiated during this project [e.g. among the already cited references: 16, 66, 17, 31]. In this project, we will collaborate with several past FoX participants, including Luc Segoufin, Diego Figueira, and Sławomir Lasota. The main originality of the present proposal is the investigation of the connections with substructural logics and proof systems.

**ANR ReacHard (2011–2014):** on *Taming hard reachability problems for counter systems* (http://www.lsv.ens-cachan.fr/Projects/anr-reachard/) is a "blanc" project coordinated by Alain Finkel at LSV, of which Sylvain Schmitz is an active participant. The project investigates the formal verification of counter systems, and in particular the complexity of reachability problems in counter systems. Many already mentioned complexity results [63, 29, 26, 28, 52, 51, 62, 24] have been obtained during the project. The computational complexity investigations of the proposal (cf. §1.2.3 and §1.3.5) are in the general continuation of ReacHard.

**ANR DynRes (2012–2014):** on *Dynamic resources and separation and update logics* (http://anr-dynres.loria.fr) is a "blanc" project coordinated by Didier Galmiche at LORIA in Nancy. It is related to the proposal, in that its participants (who include Stéphane Demri at LSV) are also interested in the model and the proof theory of substructural logics like Boolean BI and separation logic [e.g. 49] (cf. §1.2.1 and §1.2.2).

**ANR TypeX (2012–2014):** on *Typeful certified XML* (http://typex.lri.fr/) is a "blanc" project coordinated by Guiseppe Castagna at PPS in Paris. The project is closely related to our proposal in its use of formal methods and proof theory for XML processing. The project has already developed:

- Language-based techniques using typing annotations for very powerful and general XML processing languages like ℂDuce [23]. This is not directly comparable with our proposal: on one dimension, ℂDuce is of broader scope than our XPath fragments with data, but on another dimension, typing checks do not encompass satisfiability in presence of data.
- An automata-based satisfiability checking tool for navigational XPath: the *TaToo Tree Automata Toolkit* at http://git.nguyen.vg/gitweb/?p=tatoo.git. This does not solve our issues with data, but is certainly an exciting development.
- A logic-based satisfiability checking tool, also for navigational XPath: *WebSolver* at http://wam.inrialpes.fr/websolver/. This already mature tool developed over the last ten years relies on tableaux techniques for the $\mu$-calculus [37].

Broadly construed, our concerns with the verification of XPath queries with data are in the general continuation of TypeX, albeit with a very specific focus on the (especially difficult) problem of checking XPath with data.

**ANR ExStream (2014–2017):** on *Extensions of stream processing* is a young researcher project coordinated by Olivier Gauwin at LaBRI in Bordeaux that just started. Its focus is *streaming* algorithms for XML processing, i.e. where the whole data tree like the one in Figure 2b is not built in memory, but query answers are computed on-the-fly on the text document like the one in Figure 2a. Those are challenging topics, mostly orthogonal to ours.

## 2. Scientific & Technical Program

Our investigations on the relationships between data logics like XPath on the one hand and substructural logics on the other hand is a completely novel approach. We split it into three main tasks developed below—and already described up to some point in Section 1.3—, with an additional *coordination* task: Task 0. Algorithmic considerations and extensions to handle XML key constraints should permeate all three tasks.

0. Coordination and communication.
A. Proof systems for data logics.
B. Model theory.
C. Counter systems.

Before we detail the program, let us mention here some common features of the tasks, as far as the theoretical aspects of our proposal are concerned:

**Success**: The main indicator of success is publication in top venues of the field: international conferences (e.g. LICS, PODS, ICALP, ICDT, STACS, IJCAR, etc.) and journals (e.g. ACM ToCL, ACM ToDS, LMCS, MSCS, etc.), and reuse of our results by our colleagues in related fields.

The ability to extract algorithmic results, whether on a theoretical level (i.e. complexity statements for XPath fragments) or on a practical one (i.e. implemented proof search algorithms) will be a strong validation of our project.

**Risks**: Due to the challenging and sometimes prospective nature of the theoretical work we envision, it is quite possible that we will fail to find a solution, or that the solution we find will turn out to be only of mild interest. We mitigate this by always proposing some (seemingly) attainable preliminary steps for each task. Because the project lies at the intersection of the interests of several communities in database theory, proof theory, and verification, we are however confident that, should we fail to reach our exact goal, we will not run short on interesting related problems. Moreover, even partial theoretical results may provide valuable insights for the practical applications of our project.

### 2.1. **Task 0: Coordination and Communication**.

| | | |
|---|---|---|
| *Objectives.* | Management of the project. | |
| *Implication.* | Sylvain Schmitz (coordination) | |
| *Deliverables:* | | |
| $T_0 + 3$ | Project website | Website |
| $T_0 + 24$ | Intermediate report | Report |
| $T_0 + 48$ | Final project report | Report |

*Description.* This task is devoted to the organizational aspects of the project. This includes:

- creating and maintaining the project website,
- advertising for the Ph.D. position and handling applications,
- planning the yearly meetings for the project, at $T_0+3$, $T_0+12$, $T_0+24$, $T_0+36$, and $T_0+48$,
- organizing seminars and events for the dissemination of the results (we envision in particular proposing a course at ESSLLI at $T_0 + 36$),
- preparing the intermediate and final reports, and
- attending the meetings organized by the ANR.

*Risks.* There is a risk specific to Task 0, which is to fail to find any suitable Ph.D. applicant. This risk is mitigated by asking for a "long" project of four years instead of the more standard three years, thereby extending the period of search for candidates.

### 2.2. **Task A: Proof Systems for Data Logics.**

| | | |
|---|---|---|
| *Objectives.* | Develop a proof theoretic understanding of data logics | |
| *Implication.* | David Baelde (coordination), Ph.D. student, Sylvain Schmitz | |
| *Deliverables:* | | |
| $T_0 + 18$ | Benchmark suite (A.4) | Software |
| $T_0 + 24$ | Proof theory for data logics (A.1) | Document |
| $T_0 + 36$ | Automated reasoning for data logics (A.2) | Document |
| $T_0 + 48$ | Prototype (A.3) | Software |

*Description.* This is the main drift behind the project, where the ultimate goal is the development of a modular proof system for XPath with data tests, able to explain why certain fragments are decidable/tractable and others are not. It will also be the main topic of the Ph.D. student we wish to hire on the project.

Our methodology will be in two steps, as reflected into the timeline and deliverables for this task. We will first leave aside complexity or even decidability concerns, and focus solely on obtaining a clean and modular framework for understanding XPath with data. Only then will we tackle automated reasoning (in particular, satisfiability checking) in various fragments of that logic. With this plan of action, we expect to be able to provide a clearer picture of the decidability and complexity frontiers.

We present below two approaches to this program, which we believe have good chances of success. Depending on the results, either of them could serve as a basis for a prototype implementation.

*Task A.1: Modular Proof Systems for Data Logics.* So far, XPath has been given various axiomatizations and proof systems, but only for its navigational fragment. We shall study the extension of those results to cover data tests and identifiers. We will first leave aside decidability and complexity issues and focus on building a clean and modular framework accounting for the various aspects of XPath: navigation, data, and identifiers.

**Task A.1**: *Develop modular proof systems for data logics and explore their meta-theory.*

The navigational aspect, handled by means of modal logic, is pretty well understood by now [66], although only Hilbert-style axiomatizations exist. The other two aspects have not been studied so far. Reasoning on data may require some amount of first order logic and equational reasoning. We will consider here some (very weak) form of closed world equality [8] which is particularly well suited for proof search. Finally, we will borrow from hybrid logic [4] to deal with identifiers. Indeed, hybrid logics result from the extension of modal logic with *nominals* which identify world (or state), and logical connectives which allow to capture the name of the current state or jump to a state identified by a given nominal—exactly the features necessary to reflect the operations on identifiers in XPath.

The key difficulty here is to combine these three dimensions in a single logic while keeping a manageable and well structured proof system. Indeed, this task would be of little use for us if the resulting proof system was not amenable to proof search. More generally, we will strive to obtain a well behaved proof system, in (mildly extended) sequent calculus, to benefit as much as possible from the usual main results such as cut elimination, subformula property, and focused proof systems.

*Task A.2: Horn Clauses for Data Automata.* Automata can be viewed as systems of Horn clauses [35], providing an elegant formalization of alternation and two-ways navigation. The idea in this subtask is to try to lift these results to handle data automata, using additional connectives from substructural logics (cf. §1.2.1).

**Task A.2**: *Investigate data automata as systems of Horn clauses.*

They are well-known bridges between automata with Horn clauses and counter systems:

- Horn fragments of linear logic have been studied by Kanovich [45] and are directly related to counter systems; the reductions between MELL and BVASS by de Groote et al. [25] also rely on a Horn fragment of MELL as an intermediate step.
- Verma and Goubault-Larrecq [71] study various tree automata models for terms *modulo* associative and commutative symbols and relate them with counter systems (including BVASS). Counters are indeed easily encoded in unary with associative and commutative symbols.

Thus Horn clauses arise very naturally in the context of data logic.

Approaching automata through Horn clauses not only provides an elegant logical view of automata, but also leads to efficient proof search techniques by *(ordered) resolution*—we recommend the invited talk entitled “Logic Wins!” by Goubault-Larrecq [40] on the subject. See also [71] for work relating BVASS, automata, and systems of Horn clauses.

*Task A.3: Prototype Implementation.* We want to implement an XPath satisfiability checker that handles data tests in a fragment of the language.

**Task A.3**: *Provide a prototype satisfiability checker for XPath with data.*

We intend to combine proof search on the proof systems of the previous subtasks with efficient symbolic representations employed for the verification of counter machines [e.g. 56]. We leave open the exact choice for the kind of proof system we will employ to this end, and will rather evaluate the outcomes of tasks A.1 and A.2 before starting this implementation.

We will distribute the prototype as a library to ease its interfacing with model checkers for programs embedding XPath queries. Special emphasis will be given to documentation and tutorials.

*Task A.4: Benchmark Suite.* One issue we found when preparing this project proposal is the lack of a suitable benchmark suite for XPath with data. Such a benchmark suite would be highly useful in several regards:

(1) It would allow to evaluate the practical coverage of the fragments we investigate in the theoretical part of the project.
(2) It would provide a test suite for the XPath checker we propose to implement in Task A.3.
(3) It would provide a good starting point for the Ph.D. student, who would need to be confronted with practical examples to build good intuitions on the workings of XPath and data logics.

The few benchmark suites we found, like XPathMark [34], target the evaluation of XPath queries on specific inputs, rather than their static analysis independently of any given input. As a result, their coverage of data primitives varies from nonexistent to wholly insufficient. We want to constitute such a suitable benchmark.

**Task A.4**: *Compile a benchmark suite for XPath with data.*

Because XPath is embedded into other languages like XSLT or XQuery or general-purpose languages (cf. Figure 3), representative examples must be extracted and cleaned from references to the surrounding context, which considerably complicates the task.

*Risks.* A specific risk for Task A pertains to the implementation of Task A.3: the proof systems we study in tasks A.1 and A.2 might indeed be difficult to use in practice. We shall alleviate those risks:

- by keeping this concern in mind throughout the work on tasks A.1 and A.2,
- by considering *incomplete* systems as a first approximation, which often provide useful answers in practical cases—the benchmark suite of Task A.4 should be very useful in validating such simplified systems, and
- by employing over- and under-approximation techniques inspired by verification techniques for infinite-state systems [e.g. 56].

### 2.3. **Task B: Model Theory.**

|  |  |  |
|---|---|---|
| *Objectives.* | Investigate the model-theoretic relations between data and substructural logics. | |
| *Implication.* | David Baelde, Sylvain Schmitz (coordination) | |
| *Deliverables:* | | |
| $T_0 + 24$ | Data tree decomposition logic (B.1) | Document |
| $T_0 + 48$ | Proof structures as data structures (B.2) | Document |

*Description.* Whereas Task A above is concerned with the proof theoretic connections between data logics and substructural logics, this task will investigate the model-theoretic aspects of their connections. There are naturally two sides to this question, depending on which logic is used for forcing, and giving rise to two subtasks.

*Task B.1: Data Trees as Substructural Models.* This should be the most immediately useful subtask, as it will help in our other studies in tasks A and C. Building on the general intuition of data structures as resources (cf. §1.2.1), our objective is to define logics on data trees working by *decomposition.* Our aim is to define logics that are equi-expressive with some natural fragments or variants of XPath with data.

**Task B.1**: *Investigate decomposition logics for data trees.*

This entails investigating decomposition connectives for data trees, in the line of the work of Calcagno et al. [21] for trees. We can find inspiration in the various forms of multiplicative conjunctions found in e.g. linear [38], relevance [67], and abstract separation logic [22], each of which having different substructural behavior and different ways of interacting with additive connectives. The influence of key and foreign key constraints in XML documents should also be considered here, as it might involve introducing different connectives.

Algebraic notions play a critical role in the semantics for those substructural logics. Thus the task involves digesting the current state of the art on algebraic approaches for tree languages. This is a difficult topic, where the approaches through monoids and semigroups, which were so successful in the case of words, do not easily generalize to trees [13].

#### 2.3.1. *Task B.2: Proof Trees as Data Models.*

A secondary, more prospective aspect of the proposal should consider the converse direction: seeing proof structures as data structures, and attempting to characterize provability as a satisfiability problem on data logics.

**Task B.2**: *Explore substructural proof structures through data logics.*

The focus in this task will be on lambda terms describing proof structures in substructural logics. The intuition here is that, in say a linear lambda term, an abstracted variable is used once and only once. Thus a suitable coloring of linear lambda terms with data values is conceivable, and data logics on trees could be harnessed to express various properties on colored lambda terms, which means ultimately expressing properties on the proof structures associated to them through the Curry-Howard isomorphism.

Several properties should be of interest: of course, whether the data tree describes an actual proof, but also whether the proof is normalized, focused, etc. One may also attempt to characterize the restricted uses of exponentials that are behind implicitly complexity type systems guaranteeing, for instance, polynomial time normalization [39, 6, 48].

Lastly, such a characterization would provide a very simple and natural way of showing that the substructural logics defined in Task B.1 are not more powerful than required. Indeed, there would be a converse reduction from their provability problem to data logic satisfiability, using only very elementary encodings of proof trees—rather than the admittedly complex encodings of data automata etc. into substructural logics.

## 2.4. **Task C: Counter Systems.**

|  |  |  |
|---|---|---|
| *Objectives.* | Study the relations between branching counter systems and data logics on trees |  |
| *Implication.* | David Baelde, Sylvain Schmitz (coordination) |  |
| *Deliverables:* |  |  |
| $T_0 + 24$ | Branching VASS for XPath (C.2) | Document |
| $T_0 + 48$ | Data logic capturing BVASS (C.1) | Document |

*Description.* Both in the case of data logics like XPath and in that of substructural logics, various classes of counter systems have been instrumental in establishing algorithms and optimal complexity bounds [see e.g. 28, 51, 62]. Still, as explained in §1.3.3 and contrasting with the situation with data words, the exact relationships between data logics and counter systems are not known in the case of data trees. In particular, first-order logic with two variables seems to require an extension of BVASS [30]. As in the previous tasks, we present here two interesting angles to approach this question.

*Task C.1: Data logics capturing BVASS.* A first approach to this mismatch is to take BVASS as the standard branching extension of VASS, against which to define suitably restricted data logics.

**Task C.1**: *Which data logics correspond exactly to BVASS?*

Natural candidates would be logics on *disjoint* data trees (cf. §1.2.1)—but this is a semantic restriction. Could syntactic fragments of data logics have the property that, if there is a data tree model, then there is also one that is disjoint?

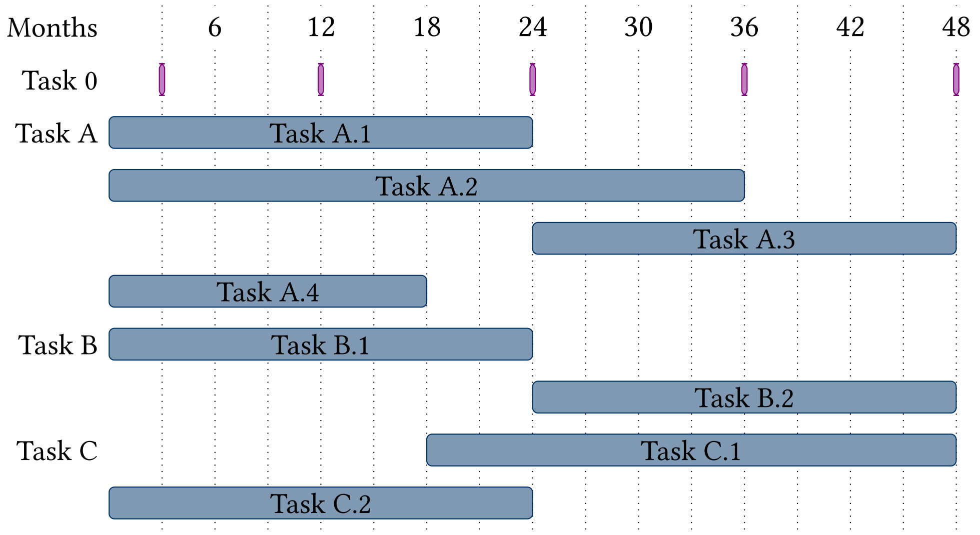
An important motivation for this subtask is that a new logical characterization (besides MELL [25]) might shed some new light on the open problem of reachability in BVASS.

*Task C.2: Data logics and fusion VASS.* The second approach is to rather adapt BVASS to better fit the intuitions behind data logics and data automata. This was done by Dimino et al. [30] for first-order logic on data trees, with a rather involved extension of BVASS. We suspect that such an extension is actually required, thus our inquiries would rather focus on the XPath case:

**Task C.2**: *What kind of branching counter systems capture XPath (fragments) with data?*

One interesting candidate is actually grounded in substructural logics: 'fusion' VASS replace the 'split' operation of BVASS, which is a strict decomposition of counter valuations—and can be seen as a multiplicative conjunction as in linear or separation logic—, by a 'fusion' operation where some uncontrolled amounts are copied—which can be seen as the conjunction of relevance logic or of the “sharing” variant of separation logic [42]. This operation seems more natural than splits in the context of data logics. As often, the case of XPath with identifier jumps might lead to an interesting mixture of behaviors.

2.5. **Calendar**. We sum up the proposed calendar for the tasks in the following diagram. The only dependencies between (sub)tasks are the dependency of Task A.3 on the proof system of either Task A.1 or Task A.2, and on the benchmark suite of Task A.4. Each subtask ends with an associated deliverable.



2.6. **Justification for the Requested Resources**. The project has a single partner: LSV, the computer science laboratory at École Normale Supérieure de Cachan. It has two permanent participants for the full four years of its duration, and a Ph.D. student for three years.

2.6.1. *Equipment: € 2,500.* We ask for two laptops together with the associated accessories and paraphernalia. One is intended for the Ph.D. student, the other one for one of the two permanent participants in replacement of aging equipment.

2.6.2. *Staff: € 98,000.* We wish to fund one Ph.D. student for three years.

2.6.3. *Missions: € 39,000.* The financial resources for the project include mission expenses in order to cover the costs of attending to conferences and of inviting external collaborators for short-term visits (ideally in conjunction with the yearly project meetings).

- Conferences: cost averaged at € 1,500 per conference. We request funding for an average of two conferences per year and participant, for a total of € 1,500 $\times$ 2 $\times$ 11 = € 33,000.
- Short-term visits: we ask for a short one-week visit for an external participant for each year of the project, for a total of € 1,500 $\times$ 4 = € 6,000.

## 3. Dissemination & Global Impact

3.1. **Dissemination**. In the first instance, the scientific results of the project will be disseminated through

- joint publications with our external collaborators in international conferences (e.g. LICS, PODS, ICALP, ICDT, STACS, IJCAR, etc.) and journals (e.g. ACM ToCL, ACM ToDS, LMCS, MSCS, etc.) of top quality, and
- through seminar and tutorial presentations in conferences and workshops.

In the last stages of the project, we will also propose a course on the themes of the project at the *European Summer School on Logic, Language and Information* (ESSLLI) to better disseminate our motivations and results to young scientists (planned at $T_0 + 36$).

We propose two main technical contributions, namely a satisfiability checker (Task A.3) and a benchmark suite (Task A.4). We will make them publicly available through open-source licenses. This will allow other teams working on the formal verification of data-aware programs to benefit from our work, and as a side-effect improve the visibility of our results.

3.2. **Impact**. The expected impact of this academic project is above all scientific. By studying the issues surrounding XPath queries and their proof systems, we will further their understanding.

Our tool and benchmark are intended to be used, evaluated, and hopefully further extended by other academic groups and research groups in the industry. On both accounts, they will fulfill a glaring void when it comes to XPath queries with data, which should help in their adoption.

## References

[1] L. Afanasiev and M. Marx. An analysis of XQuery benchmarks. *Inform. Sys.*, 33(2):155–181, 2008. doi:10.1016/j.is.2007.05.002.

[2] L. Afanasiev, P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M. Marx, and M. de Rijke. PDL for ordered trees. *J. Appl. Non-Classical Log.*, 15(2):115–135, 2005. doi:10.3166/jancl.15.115-135.

[3] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *J. Logic Comput.*, 2(3):297–347, 1992. doi:10.1093/logcom/2.3.297.

[4] C. Areces and B. ten Cate. *Hybrid logics*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 821–868. Elsevier, 2007. doi:10.1016/S1570-2464(07)80017-6.

[5] C. Areces, S. Figueira, and S. Mera. Completeness results for memory logics. *Ann. Pure App. Logic*, 163(7): 961–972, 2012. doi:10.1016/j.apal.2011.09.005.

[6] A. Asperti and L. Roversi. Intuitionistic light affine logic. *ACM Trans. Comput. Logic*, 3(1):137–175, 2002. doi:10.1145/504077.504081.

[7] D. Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Logic*, 13(1), 2012. doi:10.1145/2071368.2071370.

[8] D. Baelde and G. Nadathur. Combining deduction modulo and logics of fixed-point definitions. In *LICS 2012*, pages 105–114. IEEE, 2012. doi:10.1109/LICS.2012.22.

[9] D. Baelde, A. Gacek, D. Miller, G. Nadathur, and A. Tiu. The Bedwyr system for model checking over syntactic expressions. In *CADE 2007*, volume 4603 of *Lect. Notes in Comput. Sci.*, pages 391–397. Springer, 2007. doi:10.1007/978-3-540-73595-3_28.

[10] D. Baelde, D. Miller, and Z. Snow. Focused inductive theorem proving. In *IJCAR 2010*, volume 6173 of *Lect. Notes in Comput. Sci.*, pages 278–292. Springer, 2010. doi:10.1007/978-3-642-14203-1_24.

[11] M. Benedikt and C. Koch. XPath leashed. *ACM Comput. Surv.*, 41(1):3, 2009. doi:10.1145/1456650.1456653.

[12] M. Blockelet and S. Schmitz. Model-checking coverability graphs of vector addition systems. In *MFCS 2011*, volume 6907 of *Lect. Notes in Comput. Sci.*, pages 108–119. Springer, 2011. doi:10.1007/978-3-642-22993-0_13.

[13] M. Bojanczyk. Algebra for tree languages. Draft of a chapter for the AutomathA handbook, 2011. URL http://www.mimuw.edu.pl/~bojan/papers/treealgs.pdf.

[14] M. Bojańczyk and S. Lasota. An extension of data automata that captures XPath. In *LICS 2010*, pages 243–252. IEEE, 2010. doi:10.1109/LICS.2010.33.

[15] M. Bojańczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3):1–48, 2009. doi:10.1145/1516512.1516515.

[16] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Logic*, 12(4):27:1–27:26, 2011. doi:10.1145/1970398.1970403.

[17] M. Bojańczyk, B. Klin, and S. Lasota. Automata with group actions. In *LICS 2011*, pages 355–364. IEEE, 2011. doi:10.1109/LICS.2011.48.

[18] A. Bouajjani and M. Emmi. Analysis of recursively parallel programs. *ACM Trans. Prog. Lang. Syst.*, 35 (3):10:1–10:49, 2013. doi:10.1145/2518188.

[19] J. Brotherston. Bunched logics displayed. *Studia Logica*, 100(6):1223–1254, 2012. doi:10.1007/s11225-012-9449-0.

[20] J. Brotherston and M. Kanovich. Undecidability of propositional separation logic and its neighbours. *J. ACM*, 61(2), 2014. doi:10.1145/2542667.

[21] C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding validity in a spatial logic for trees. *J. Funct. Programming*, 15(4):543–572, 2005. doi:10.1017/S0956796804005404.

[22] C. Calcagno, P. W. O'Hearn, and H. Yang. Local action and abstract separation logic. In *LICS 2007*, pages 366–378. IEEE, 2007. doi:10.1109/LICS.2007.30.

[23] G. Castagna, K. Nguyen, Z. Xu, H. Im, S. Lenglet, and L. Padovani. Polymorphic functions with set-theoretic types: Part 1: Syntax, semantics, and evaluation. In *POPL 2014*, pages 5–17. ACM, 2014. doi:10.1145/2535838.2535840.

[24] J.-B. Courtois and S. Schmitz. Alternating vector addition systems with states. Manuscript, Apr. 2014. URL http://hal.inria.fr/hal-00980878.

[25] Ph. de Groote, B. Guillaume, and S. Salvati. Vector addition tree automata. In *LICS 2004*, pages 64–73. IEEE, 2004. doi:10.1109/LICS.2004.51.

[26] S. Demri. On selective unboundedness of VASS. *J. Comput. Syst. Sci.*, 79(5):689–713, 2013. doi:10.1016/j.jcss.2013.01.014.

[27] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Logic*, 10 (3):16, 2009. doi:10.1145/1507244.1507246.

[28] S. Demri, D. Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. In *LICS 2013*, pages 33–42. IEEE, 2013. doi:10.1109/LICS.2013.8.

[29] S. Demri, M. Jurdziński, O. Lachish, and R. Lazić. The covering and boundedness problems for branching vector addition systems. *J. Comput. Syst. Sci.*, 79(1): 23–38, 2013. doi:10.1016/j.jcss.2012.04.002.

[30] J. Dimino, F. Jacquemard, and L. Segoufin. $\mathrm{FO}^2(<, +1, \sim)$ on data trees, data tree automata and an extension of BVASS. Manuscript, hal.inria.fr:hal-00769249, 2013.

[31] D. Figueira. Alternating register automata on finite words and trees. *Logic. Meth. in Comput. Sci.*, 8(1):22, 2012. doi:10.2168/LMCS-8(1:22)2012.

[32] D. Figueira. On XPath with transitive axes and data tests. In *PODS 2013*, pages 249–260. ACM, 2013. doi:10.1145/2463664.2463675.

[33] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256 (1–2):63–92, 2001. doi:10.1016/S0304-3975(00)00102-X.

[34] M. Franceschet. XPathMark – an XPath benchmark for XMark generated data. In *XSYM 2005*, volume 3671 of *Lect. Notes in Comput. Sci.*, pages 129–143, 2005. doi:10.1007/11547273_10.

[35] T. W. Frühwirth, E. Y. Shapiro, M. Y. Vardi, and E. Yardeni. Logic programs as types for logic programs. In *LICS 1991*, pages 300–309. IEEE, 1991. doi:10.1109/LICS.1991.151654.

[36] P. Genevès and J.-Y. Vion-Dury. Logic-based XPath optimization. In *DocEng 2004*, pages 211–219. ACM, 2004. doi:10.1145/1030397.1030437.

[37] P. Genevès, N. Layaïda, and A. Schmitt. Efficiently deciding mu-calculus with converse over finite trees. 2013. URL http://hal.inria.fr/hal-00868722.

[38] J.-Y. Girard. Linear logic. *Theor. Comput. Sci.*, 50(1):1–101, 1987. doi:10.1016/0304-3975(87)90045-4.

[39] J.-Y. Girard. Light linear logic. *Inform. and Comput.*, 143(2):175–204, 1998. doi:10.1006/inco.1998.2700.

[40] J. Goubault-Larrecq. "Logic wins!". In *Asian 2009*, volume 5913 of *Lect. Notes in Comput. Sci.*, pages 1–16. Springer, 2009. doi:10.1007/978-3-642-10622-4_1.

[41] J. Groppe and S. Groppe. A prototype of a schema-based XPath satisfiability tester. In *DEXA 2006*, volume 4080 of *Lect. Notes in Comput. Sci.*, pages 93–103. Springer, 2006. doi:10.1007/11827405_10.

[42] A. Hobor and J. Villard. The ramifications of sharing in data structures. In *POPL 2012*, pages 523–536. ACM, 2012. doi:10.1145/2429069.2429131.

[43] Z. Hóu, A. Tiu, and R. Goré. A labelled sequent calculus for BBI: Proof theory and proof search. In *Tableaux 2013*, volume 8123 of *Lect. Notes in Comput. Sci.*, pages 172–187. Springer, 2013. doi:10.1007/978-3-642-40537-2_16.

[44] M. Jurdziński and R. Lazić. Alternating automata on data trees and XPath satisfiability. *ACM Trans. Comput. Logic*, 12(3), 2011. doi:10.1145/1929954.1929956.

[45] M. I. Kanovich. Petri nets, Horn programs, linear logic and vector games. *Ann. Pure App. Logic*, 75(1–2):107–135, 1995. doi:10.1016/0168-0072(94)00060-G.

[46] A. P. Kopylov. Decidability of linear affine logic. *Inform. and Comput.*, 164(1):173–198, 2001. doi:10.1006/inco.1999.2834.

[47] S. A. Kripke. The problem of entailment. In *ASL 1959*, volume 24 of *J. Symb. Log.*, page 324, 1959. doi:10.2307/2963903. Abstract.

[48] Y. Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1):163–180, 2004. doi:10.1016/j.tcs.2003.10.018.

[49] D. Larchey-Wendling and D. Galmiche. Nondeterministic phase semantics and the undecidability of Boolean BI. *ACM Trans. Comput. Logic*, 14(1):6:1–6:41, 2013. doi:10.1145/2422085.2422091.

[50] R. Lazić. The reachability problem for branching vector addition systems requires doubly-exponential space. *Inf. Process. Lett.*, 110(17):740–745, 2010. doi:10.1016/j.ipl.2010.06.008.

[51] R. Lazić and S. Schmitz. Non-elementary complexities for branching VASS, MELL, and extensions. In *CSL-LICS 2014*. ACM, 2014. doi:10.1145/2603088.2603129. To appear, arXiv:1401.6785 [cs.LO].

[52] J. Leroux, M. Praveen, and G. Sutre. A relational trace logic for vector addition systems with application to context-freeness. In *Concur 2013*, volume 8052 of *Lect. Notes in Comput. Sci.*, pages 137–151. Springer, 2013. doi:10.1007/978-3-642-40184-8_11.

[53] C. Liang and D. Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theor. Comput. Sci.*, 410(46):4747–4768, 2009. doi:10.1016/j.tcs.2009.07.041.

[54] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. *Ann. Pure App. Logic*, 56(1–3):239–311, 1992. doi:10.1016/0168-0072(92)90075-B.

[55] R. Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University, 1976.

[56] R. Majumdar and Z. Wang. Expand, enlarge, and check for branching vector addition systems. In *Concur 2013*, volume 8052 of *Lect. Notes in Comput. Sci.*, pages 152–166. Springer, 2013. doi:10.1007/978-3-642-40184-8_12.

[57] V. Nigam and D. Miller. Algorithmic specifications in linear logic with subexponentials. In *PPDP 2009*, pages 129–140. ACM, 2009. doi:10.1145/1599410.1599427.

[58] P. W. O'Hearn and D. J. Pym. The logic of bunched implications. *Bull. Symb. Log.*, 5(2):215–244, 1999. doi:10.2307/421090.

[59] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6(2):223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

[60] O. Rambow. Multiset-valued linear index grammars: imposing dominance constraints on derivations. In *ACL'94*, pages 263–270. ACL Press, 1994. doi:10.3115/981732.981768.

[61] S. Schmitz. On the computational complexity of dominance links in grammatical formalisms. In *ACL 2010*, pages 514–524. ACL Press, 2010. URL http://hal.archives-ouvertes.fr/hal-00482396.

[62] S. Schmitz. Implicational relevance logic is 2-ExpTime-complete. In *RTA-TLCA 2014*, Lect. Notes in Comput. Sci. Springer, 2014. To appear, arXiv:1402.0705 [cs.LO].

[63] S. Schmitz and Ph. Schnoebelen. Algorithmic aspects of wqo theory. Lecture notes, 2012. URL http://cel.archives-ouvertes.fr/cel-00727025.

[64] Ph. Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *MFCS 2010*, volume 6281 of *Lect. Notes in Comput. Sci.*, pages 616–628. Springer, 2010. doi:10.1007/978-3-642-15155-2_54.

[65] B. ten Cate and C. Lutz. The complexity of query containment in expressive fragments of XPath 2.0. *J. ACM*, 56(6):31:1–31:48, 2009. doi:10.1145/1568318.1568321.

[66] B. ten Cate, T. Litak, and M. Marx. Complete axiomatizations for XPath fragments. *J. Appl. Logic*, 8(2):153–172, 2010. doi:10.1016/j.jal.2009.09.002.

[67] A. Urquhart. Semantics for relevant logics. *J. Symb. Log.*, 37(1):159–169, 1972. doi:10.2307/2272559.

[68] A. Urquhart. The undecidability of entailment and relevant implication. *J. Symb. Log.*, 49(4):1059–1073, 1984. doi:10.2307/2274261.

[69] A. Urquhart. The complexity of decision procedures in relevance logic II. *J. Symb. Log.*, 64(4):1774–1802, 1999. doi:10.2307/2586811.

[70] K. N. Verma and J. Goubault-Larrecq. Karp-Miller trees for a branching extension of VASS. *Disc. Methods in Theor. Comput. Sci.*, 7(1):217–230, 2005. URL http://www.dmtcs.org/volumes/abstracts/dm070113.abs.html.

[71] K. N. Verma and J. Goubault-Larrecq. Alternating two-way AC-tree automata. *Inform. and Comput.*, 205(6):817–869, 2007. doi:10.1016/j.ic.2006.12.006.