Reasoning on words and trees with data
On decidable automata on data words and data trees in relation to satisfiability of LTL and XPath.

Diego Figueira
verification of databases

return all the circles that have the *same color* as a star

query → database → result set =
verification of databases

\[ q(\text{database}) = \{\text{result set}\} \]
verification of databases

\[ q(\text{database}) = \{ \text{result set} \} \]

query database result set

static analysis
verification of databases

\[ q(\text{query}) = \begin{cases} \text{result set} \\ \exists \quad q(\text{query}) \neq \emptyset ? \end{cases} \]

static analysis

does \( q \) express a property?
verification of databases

\[ q(\text{database}) = \{ \text{result set} \} \]

query \quad database

\[ \forall q(\text{database}) = q'(\text{database}) ? \]

static analysis

equivalence

can we use \( q' \) instead of \( q \)?

satisfiability

does \( q \) express a property?
verification of databases

query database result set

\[ q(\text{database}) = \{ \text{result set} \} \]

query database result set

\[ \forall \subseteq \] static analysis

satisfiability does \( q \) express a property?

containment are \( q \) and \( q' \) comparable queries?
verification of databases

\[ q(\text{\textbullet}) = \{ \text{result set} \} \]

query database

static analysis

equivalence

can we use \( q' \) instead of \( q \)?

satisfiability

does \( q \) express a property?

containment

are \( q \) and \( q' \) comparable queries?
databases with structure

relational databases
databases with structure

relational databases

tree-like structure
unranked, ordered finite trees
XML documents
“data tree”
databases with structure

relational databases

databases with structure

relational databases

tree-like structure
unranked, ordered finite trees
XML documents
“data tree”
databases with structure

relational databases

databases with structure

A B C

C A B

? ? ?

databases with structure

tree-like structure
unranked, ordered finite trees
XML documents
“data tree”

databases with structure

special case of a data tree
temporal databases
“data word”

databases with structure

relational databases

databases with structure

A B C

C A B

? ? ?

databases with structure

tree-like structure
unranked, ordered finite trees
XML documents
“data tree”

databases with structure

special case of a data tree
temporal databases
“data word”
databases with structure

- relational databases
- tree-like structure
  - unranked, ordered finite trees
  - XML documents
    - “data tree”

special case of a data tree
- temporal databases
  - “data word”
data values
work on words and trees with
- finite alphabet
- structure
data values
work on words and trees with
- finite alphabet
- structure

need to perform joins,
to test for equality of data values
data values
work on words and trees with
- finite alphabet
- structure

“return all products in stock”

need to perform joins,
to test for equality of data values
data values
work on words and trees with
- finite alphabet
- structure

"return all products in stock"

need to perform joins,
to test for equality of data values
what's out there?

(alternating) register automata
Alur, Dill, . . .

pebble automata
Kaminski, Francez, Neven, Schwentick, Vianu, Tan, Demri, Lazić, Jurdziński, . . .

≈
[F, Hofman, Lasota, EXPRESS'10]

(alternating) timed automata
Alur, Dill, . . .

FO²
Bojańczyk, Muscholl, Schwentick, Segoufin, David, . . .

XPath
Benedikt, Fan, Geerts

patterns
conjunctive queries
David, Björklund, Martens, Schwenttick

LTL with registers
Demri, Lazić, Nowak

real time logics
Alur, Henzinger, Harel, Lichtenstein, Pnueli, . . .

Hybrid logics
Description Logics with concrete domains
what's out there?

(alternating) register automata
pebble automata
Kaminski, Francez, Neven, Schwentick, Vianu, Tan, Demri, Lazić, Jurdziński, ... 

[≈]
[F, Hofman, Lasota, EXPRESS'10]

(alternating) timed automata
Alur, Dill, ... 

LTL with registers
Demri, Lazić, Nowak 

XPath
Benedikt, Fan, Geerts 

FO²
Bojańczyk, Muscholl, Schwentick, Segoufin, David, ... 

patterns
conjunctive queries
David, Björklund, Martens, Schwentick 

real time logics
Alur, Henzinger, Harel, Lichtenstein, Pnueli, ... 

Hybrid logics
Description Logics with concrete domains 

hybrid logics
objective

data-aware languages & automata for word-tree-structured dbs
objective

data-aware languages & automata for word-& tree-structured dbs

rLTL data-words

XPath data-trees
objective

data-aware languages\textsubscript{\&} automata for word\textsubscript{\&} tree-structured dbs

\texttt{rLTL} data-words $\simeq$ $\neq$ XPath data-trees
Objective

data-aware languages & automata for word- & tree-structured dbms

rLTL data-words \equiv \neq XPath data-trees

equivalence satisfiability inclusion
objective

data-aware languages & automata for word-tree-structured dbs

rLTL data-words = XPath data-trees

equivalence ≠ satisfiability inclusion
Objective

Data-aware languages & automata for word & tree-structured databases

rLTL data-words

XPath data-trees

Equivalence

Satisfiability

Inclusion
rLTL\((F,X)\)

register-LTL over data words

\( \phi ::= a \mid F\phi \mid X\phi \mid \phi \lor \phi \mid \phi \land \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \)

tests current datum against the stored
stores current datum

e.g.

“For every a there is a future b with the same value.”

\[ G ( \neg a \lor \downarrow F (b \land \uparrow) ) \]
lower bounds

We knew...
[Demri / Lazić]

rLTL(X, F) is decidable, non primitive recursive

rLTL(X, F, F^{-1}) is undecidable
lower bounds

We knew... Now we know... The one-step is not necessary to obtain the lower-bounds!

rLTL(X, F) is decidable, non primitive recursive

rLTL(X, F, F⁻¹) is undecidable
lower bounds

We knew...
[Demri / Lazić]

Now we know ...

The one-step is not necessary to obtain the lower-bounds!

\[ \text{rLTL}(X, F) \text{ is decidable, non primitive recursive} \]

\[ \text{rLTL}(X, F, F^{-1}) \text{ is undecidable} \]

even for a very simple fragment

[F, Segoufin, MFCS’09]
\( rLTL(U, X) \)

register-LTL over data words

\[ \phi ::= a \mid \phi U \phi \mid X \phi \mid \phi \lor \phi \mid \phi \land \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \]

tests current datum against the stored

stores current datum

e.g.
\(\text{rLTL}(U, X)\)

register-LTL over data words

\[ \phi ::= a \mid \phi U \phi \mid X\phi \mid \phi \lor \phi \mid \phi \land \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \mid \forall \phi \mid \exists \geq \phi \]

- Exists a future data value s.t. \(\phi\).
- For all past data value, \(\phi\).
\( rLTL(U, X) \)

register-LTL over data words

\[ \phi ::= a \mid \phi U \phi \mid X \phi \mid \phi \lor \phi \mid \phi \land \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \mid \forall \leq \phi \mid \exists \geq \phi \]

- Exists a future data value s.t. \( \phi \).
- For all past data value, \( \phi \).

\[ (U, X) \]

e.g.

"There is a \( b \) with a data value different from any previous element."

\[ F( \neg b \lor \forall \leq (\neg \uparrow) ) \]

\[ \begin{array}{cccccccc}
a, 1 & \text{b, 1} & \text{a, 3} & \text{b, 1} & \text{c, 7} & \text{a, 7} & \text{a, 1} & \text{b, 5} & \text{b, 3} & \text{c, 2} & \text{b, 7} \end{array} \]
The image describes a formal logic system called rLTL(U,X), which is register-LTL over data words. The system includes the following operators:

- $\phi ::= a \mid \phi U \phi \mid X \phi \mid \phi \lor \phi \mid \phi \land \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \phi \mid \forall \leq \phi \mid \exists \geq \phi$

- For all past data value, $\phi$.
- Exists a future data value s.t. $\phi$.

An example is given:

- “There exists a data value that is contained in a $b$ but not in an $a$.”

- $\exists \geq ( F (b \land \uparrow) \land \neg F (a \land \uparrow) )$

The diagram illustrates a sequence of data values with transitions between them, demonstrating the application of the rLTL(U,X) logic system.
The satisfiability problem for positive \( \text{rLTL}(U,X) + \forall \leq + \exists \geq \) is \textit{decidable}. 

\[
\text{rLTL}(U,X)
\]

\( \phi ::= a \mid \phi U \phi \mid X \phi \mid \phi \lor \phi \mid \phi \land \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \phi \mid \forall \leq \phi \mid \exists \geq \phi \)
The satisfiability problem for positive $\text{rLTL}(U,X) + \forall \leq + \exists \geq$ is **decidable**.

The satisfiability problem for $\text{rLTL}(U,X) + \forall \leq + \exists \geq$ is **undecidable**.
ARA \[\text{Decidable emptiness problem}\] \[\text{With non-primitive recursive complexity}\] \[\text{Closed under complementation, intersection, union}\]
**ARA**

[Demri / Lazić]

- ☀ Decidable emptiness problem
- ☹ With non-primitive recursive complexity
- ☀ Closed under complementation, intersection, union

---

**ARA + guess + spread**

- ☀ Still decidable emptiness problem
- ☹ No longer closed under complementation
- ☠ Can't be closed under complementation preserving decidability
ARA \[\xrightarrow{\text{opa}}\] \text{satisfiability of } r\text{LTL}(U,X) \\

[Demri / Lazić]

- 😊 Decidable emptiness problem
- 😞 With non-primitive recursive complexity
- 😊 Closed under complementation, intersection, union

ARA + guess + spread \[\xleftarrow{\text{opa}}\] \text{satisfiability of positive } r\text{LTL}(U,X) + \forall \leq + \exists \geq

- 😊 Still decidable emptiness problem
- 😞 No longer closed under complementation
- ☠ Can't be closed under complement preserving decidability
Alternating register automata
one-way
1 register

transitions

states: 
initial state: 
final states: 

1 5 3 1 5 6 8 3 2 3
Alternating register automata
one-way
1 register

states:
initial state:
final states:

transitions

data values
A configuration = a set of running threads (state, datum)

the string is $(ab)^*$, and all the $a$'s have different data values
A configuration = a set of running threads (state, datum)

the string is \((ab)^*\), and all the a's have different data values
A configuration = a set of running threads (state, datum)

the string is \((ab)^*\), and all the \(a\)'s have different data values
examples

“There exists a data value that is contained in a \textit{b} but not in an \textit{a}.”

1. Guess a data value $d$
2. Create two threads that test
   a. there is a future position $b$ with datum $d$
   b. there is not a future position $a$ with datum $d$
examples

“There exists a data value that is contained in a b but not in an a.”
1. Guess a data value d
2. Create two threads that test
   a. there is a future position b with datum d
   b. there is not a future position a with datum d

“There is a b with a data value different from any previous element.”
Create two threads
1. One that at every position create a thread that
   a. stores current data value, and
   b. advances until the last position using some state q
2. Other that
   a. guesses a b position of the word
   b. spreads all threads with q to threads with p (state p checks that the stored data value is different from the current value)
XPath

node expressions

\[ \varphi, \psi ::= a \mid \neg \varphi \mid \varphi \land \psi \mid \alpha=\beta \mid \alpha \neq \beta \mid a? \quad a \in A \]

path expressions

\[ \alpha, \beta ::= \varepsilon \mid \alpha \beta \mid \alpha \cup \beta \mid \alpha[\varphi] \mid o \]

\[ o \in \{ \downarrow, \downarrow, \rightarrow, \cdots, \uparrow, \hat{\uparrow}, \leftarrow, \cdots \} \]

'→' : one-step

'⋯→' : multistep (transitive closure of '→')
XPath

node expressions

\( \phi, \psi ::= a \mid \neg \phi \mid \phi \land \psi \mid a = \beta \mid a \neq \beta \mid \alpha ? \mid a \in A \)

denote sets of nodes

path expressions

\( \alpha, \beta ::= \epsilon \mid \alpha \beta \mid \alpha \cup \beta \mid \alpha[\phi] \mid o \)

\( o \in \{ \downarrow, \downarrow, \rightarrow, \bullet \bullet \rightarrow, \uparrow, \uparrow, \leftarrow, \bullet \bullet \leftarrow \} \)

denote binary relations

\( \rightarrow \) : one-step

\( \bullet \bullet \rightarrow \) : multistep (transitive closure of \( \rightarrow \))
XPath

We note “XPath(=,→,⋯,↑)”
	node expressions

\[ \varphi, \psi ::= a \mid \neg \varphi \mid \varphi \land \psi \mid \alpha = \beta \mid \alpha \neq \beta \mid \alpha^? \quad a \in A \]

denote sets of nodes

path expressions

\[ \alpha, \beta ::= \varepsilon \mid \alpha \beta \mid \alpha \cup \beta \mid \alpha[\varphi] \mid o \]

\[ o \in \{ \downarrow, \downarrow, \rightarrow, \rightarrow, \uparrow, \uparrow, \leftarrow, \leftarrow, \cdots \} \]

denote binary relations

‘→’ : one-step

‘⋯’ : multistep (transitive closure of ‘→’)
There are an $a$ and a $b$ with the same datum

\[\downarrow [a] = \downarrow [b]\]
There are an \textit{a} and a \textit{b} with the same datum.

There are two \textit{c} with different data values.
\[\downarrow [a] = \downarrow [b]\]

\[\downarrow [c] \neq \downarrow [c]\]

\[\neg (\downarrow [b] \neq \downarrow [b])\]

There are an \(a\) and a \(b\) with the same datum

There are two \(c\) with different data values

All \(b\) have the same data value
There are an \( a \) and a \( b \) with the same datum

There are two \( c \) with different data values

All \( b \) have the same data value

There is no data value shared by a \( b \) and a \( c \)
There are an \( a \) and a \( b \) with the same datum

There are two \( c \) with different data values

All \( b \) have the same data value

There is no data value shared by a \( b \) and a \( c \)

The \( c \) constitutes a primary key
There are an \( a \) and a \( b \) with the same datum

There are two \( c \) with different data values

All \( b \) have the same data value

There is no data value shared by a \( b \) and a \( c \)

The \( c \) constitutes a primary key

Every \( a \) has an ancestor \( b \) with different value
satisfiability of XPath

Satisfiability of XPath (=, \(\downarrow\), \(\downarrow\), →, ⋯, ↑, ⬆, ←, ⋯) is undecidable.
satisfiability of XPath

Satisfiability of XPath(=, $\downarrow$, $\downarrow\downarrow$, $\rightarrow$, $\bullet$, $\uparrow$, $\hat{\downarrow}$, $\leftarrow$, $\cdots$) is undecidable.
satisfiability of XPath

Satisfiability of XPath(=, $\downarrow$, $\uparrow$, $\rightarrow$, $\leftarrow$, $\rightarrow\leftarrow$, $\leftarrow\rightarrow$) is undecidable.
satisfiability of XPath

Our contribution:

Satisfiability of $\mathsf{XPath}(=, \downarrow, \downarrow, \rightarrow, \cdots, \uparrow, \uparrow, \leftarrow, \cdots)$ is undecidable. [F, Segoufin, MFCS’09]
satisfiability of XPath

Our contribution:

Satisfiability of $\textbf{XPath}(=, \downarrow, \downarrow., \rightarrow, \cdots, \uparrow, \uparrow., \leftarrow, \cdots)$ is \textit{undecidable}. [F, Segoufin, MFCS’09]
satisfiability of XPath

Our contribution:

Satisfiability of $\text{XPath}(=, \downarrow, \uparrow,\rightarrow,\cdots,\hat{\uparrow},\hat{\downarrow},\cdots)$ is undecidable. [F, Segoufin, MFCS’09]

Satisfiability of $\text{XPath}(=, \downarrow, \hat{\downarrow})$ is ExpTime-complete. [F, PODS’09]
satisfiability of XPath

Our contribution:

Satisfiability of $\text{XPath}(=, \downarrow, \Downarrow, \rightarrow, \cdots, \uparrow, \Uparrow, \leftarrow, \cdots)$ is undecidable. [F, Segoufin, MFCS’09]

Satisfiability of $\text{XPath}(=, \downarrow, \Downarrow)$ is ExpTime-complete. [F, PODS’09]

Satisfiability of $\text{XPath}(=, \downarrow, \Downarrow, \rightarrow, \cdots)$ is decidable. [F, ICDT’10]
satisfiability of XPath

Our contribution:

Satisfiability of $\text{XPath}(=, \downarrow, \downarrow, \rightarrow, \leftarrow, \uparrow, \hat{\uparrow}, \hat{\rightarrow}, \hat{\leftarrow})$ is undecidable. [F, Segoufin, MFCS’09]

Satisfiability of $\text{XPath}(=, \downarrow, \downarrow)$ is ExpTime-complete. [F, PODS’09]

Satisfiability of $\text{XPath}(=, \downarrow, \downarrow, \rightarrow, \leftarrow)$ is decidable. [F, ICDT’10]

Satisfiability of $\text{XPath}(=, \downarrow, \downarrow, \uparrow, \hat{\uparrow})$ is decidable. [F, Segoufin, STACS’11]
satisfiability of XPath

Our contribution:

Satisfiability of $\text{XPath} (=, \downarrow, \downarrow, \rightarrow, \cdots, \uparrow, \uparrow, \leftarrow, \cdots)$ is undecidable. [F, Segoufin, MFCS’09]

Satisfiability of $\text{XPath} (=, \downarrow, \downarrow)$ is ExpTime-complete. [F, PODS’09]

Satisfiability of $\text{XPath} (=, \downarrow, \downarrow, \rightarrow, \cdots)$ is decidable. [F, ICDT’10]

Satisfiability of $\text{XPath} (=, \downarrow, \downarrow, \uparrow, \uparrow)$ is decidable. [F, Segoufin, STACS’11]
forward-XPath

Satisfiability of $\text{XPath}(=, \downarrow, \hat{\downarrow} \rightarrow, \cdots)$
Satisfiability of $\varepsilon$-XPath ($=, \downarrow, \lhook\rightarrow, \cdots$) is decidable. [Lazić, Jurdziński, ‘07]

only tests: $\varepsilon = \alpha$
Satisfiability of $\varepsilon\text{-XPath}(=, \downarrow, \leftarrow, \rightarrow)$ is decidable. 

[Lazić, Jurdziński, ‘07] only tests: $\varepsilon = \alpha$

We show

Satisfiability of $\text{XPath}(=, \downarrow, \leftarrow, \rightarrow)$ is decidable. 

[F, ICDT’10] lower bound: non primitive recursive
Satisfiability of $\varepsilon$-XPath(=, ↓, ⇑→,⋯) is decidable.

only tests: $\varepsilon = \alpha$

We show

Satisfiability of XPath(=, ↓, ⇑→,⋯) is decidable.

lower bound: non primitive recursive

forward XPath $\Rightarrow$ ATRA $+$ guess $+$ spread

extension of ARA on trees
alternating automata
1 register
top-down, unranked
decidable emptiness pb
forward-XPath

Satisfiability of $\varepsilon$-XPath(=, $\downarrow$, $\downarrow\uparrow\rightarrow$, $\cdots$) is decidable.

[Lazić, Jurdziński, '07] only tests: $\varepsilon = \alpha$

We show

Satisfiability of XPath(=, $\downarrow$, $\downarrow\uparrow\rightarrow$, $\cdots$) is decidable.

[F, ICDT'10] lower bound: non primitive recursive

forward XPath $\rightarrow$ ATRA + guess + spread

difficulty: ATRA are not closed under complementation

extension of ARA on trees
alternating automata
1 register
top-down, unranked
decidable emptiness pb
how do we code...?

There are an $a$ and a $b$ with the same datum

$\downarrow [a] = \downarrow [b]$
how do we code...?

\[\downarrow [a] = \downarrow [b]\]

There are an \textit{a} and a \textit{b} with the same datum

Guess the data value 7.
how do we code...?

\[ \downarrow [a] = \downarrow [b] \]

Guess the data value 7.

Check that it can be accessed with “\( \downarrow [b] \)”. 

There are an \( a \) and a \( b \) with the same datum.
how do we code...?

There are an $a$ and a $b$ with the same datum

\[ \downarrow [a] = \downarrow [b] \]

Guess the data value 7.

Check that it can be accessed with “$\downarrow [b]$”.

Check that it can be accessed with “$\downarrow [a]$”.

\[
\begin{array}{llllllllllll}
\text{a} & 3 & & & & & & & & & & \downarrow [a] = \downarrow [b] & \\
\text{b} & 7 & & & & & & & & & & & \\
\text{c} & 1 & & & & & & & & & & & \\
\text{a} & 1 & 1 & & & & & & & & & & \\
\text{b} & 7 & & & & & & & & & & & \\
\text{a} & 6 & & & & & & & & & & & \\
\text{a} & 1 & & & & & & & & & & & \\
\text{a} & 6 & & & & & & & & & & & \\
\text{a} & 1 & & & & & & & & & & & \\
\text{c} & 3 & & & & & & & & & & & \\
\text{a} & 7 & & & & & & & & & & & \\
\end{array}
\]
how do we code...?

\[ \Downarrow [a] = \Downarrow [b] \]

\[ \Downarrow [c] \neq \Downarrow [c] \]

\[ \neg ( \Downarrow [b] \neq \Downarrow [b] ) \]

There are an \(a\) and a \(b\) with the same datum.

There are two \(c\) with different data values.

All \(b\) have the same data value.
how do we code...?

There are an \( a \) and a \( b \) with the same datum

There are two \( c \) with different data values

All \( b \) have the same data value

There is no data value shared by a \( b \) and a \( c \)
how do we code...?

There are an $a$ and a $b$ with the same datum

There are two $c$ with different data values

All $b$ have the same data value

There is no data value shared by a $b$ and a $c$

build automaton $A$ such that

if $T \models \phi$, then $A$ accepts $T$

if $A$ accepts $T$, then there is $T' = f(A, T)$ such that $T' \models \phi$
XPath: inclusion\&equivalence of *path expressions*
future work

XPath: inclusion&equivalence of *path expressions*

More domain specific operations

number order arithmetic
string substring
prefix
future work

XPath: inclusion\&equivalence of path expressions

More domain specific operations

Quest for an expressive formalism with low complexity
future work

XPath: inclusion\&equivalence of *path expressions*

More domain specific operations

Quest for an expressive formalism with *low complexity*

restricting the logics/automata
restricting the class of models to obtain decidable/tractable procedures
Results

1. Horizontal

rLTL + ∀≤ + ∃≥ is decidable
ARA(guess, spread) is decidable
rLTL(F) is non primitive recursive
rLTL(F, F⁻¹) is undecidable
Results

1. Horizontal

- rLTL + ∀≤ + ∃≥ is decidable
- ARA(guess, spread) is decidable
- rLTL(F) is non primitive recursive
- rLTL(F, F^{-1}) is undecidable

2. Downward

- XPath(=, ↓, ⩾) is ExpTime-complete
- DD automaton is 2ExpTime
## Results

### 1. Horizontal

- rLTL + $\forall \leq + \exists \geq$ is decidable
- ARA(guess, spread) is decidable
- rLTL(F) is non primitive recursive
- rLTL(F, F^{-1}) is undecidable

### 2. Downward

- XPath(=, ↓, ⩾) is ExpTime-complete
- DD automaton is 2ExpTime

### 3. Forward

- XPath(=, ↓, ⩾, →, ⋯) is decidable, non primitive recursive
- ATRA(guess, spread) is decidable
Results

1. Horizontal
   - rLTL + ∀≤ + ∃≥ is decidable
   - ARA(guess, spread) is decidable
   - rLTL(F) is non primitive recursive
   - rLTL(F, F⁻¹) is undecidable

2. Downward
   - XPath(=, ↓, ⩾) is ExpTime-complete
   - DD automaton is 2ExpTime

3. Forward
   - XPath(=, ↓, ⩾, →, ⩾) is decidable, non primitive recursive
   - ATRA(guess, spread) is decidable

4. Vertical
   - XPath(=, ↓, ⩾, ↑, ⩾) is decidable, non primitive recursive
   - BUDA is decidable
Results

1. Horizontal
   - rLTL + ∀≤ + ∃≥ is decidable
   - ARA(guess, spread) is decidable
   - rLTL(F) is non primitive recursive
   - rLTL(F, F⁻¹) is undecidable

2. Downward
   - XPath(=, ↓, ⪯) is ExpTime-complete
   - DD automaton is 2ExpTime

3. Forward
   - XPath(=, ↓, ⪯, →, ···) is decidable, non primitive recursive
   - ATRA(guess, spread) is decidable

4. Vertical
   - XPath(=, ↓, ⪯, ↑, ⬆) is decidable, non primitive recursive
   - BUDA is decidable