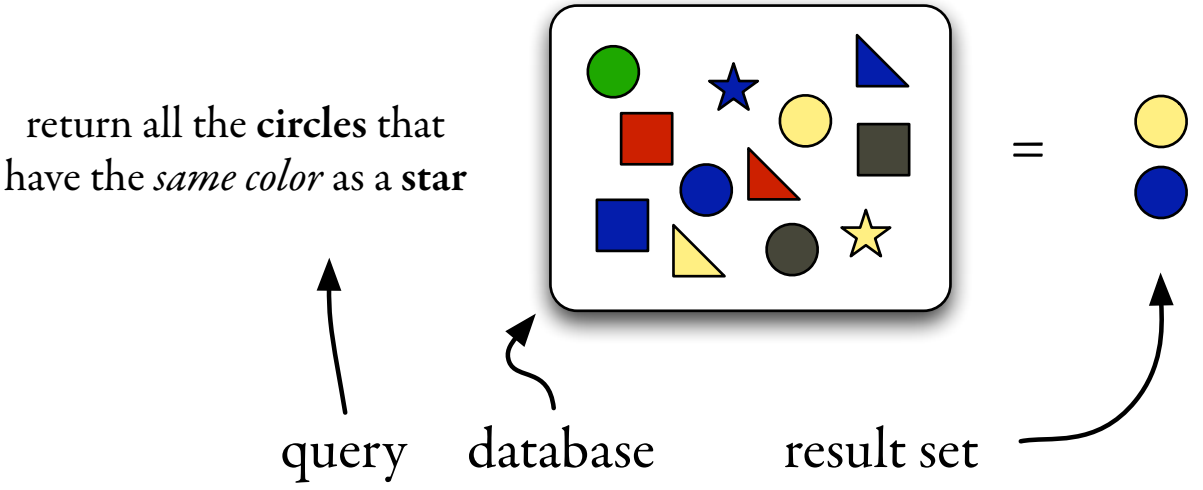
06-Dec-2010
Cachan, France

Reasoning on words and trees with data

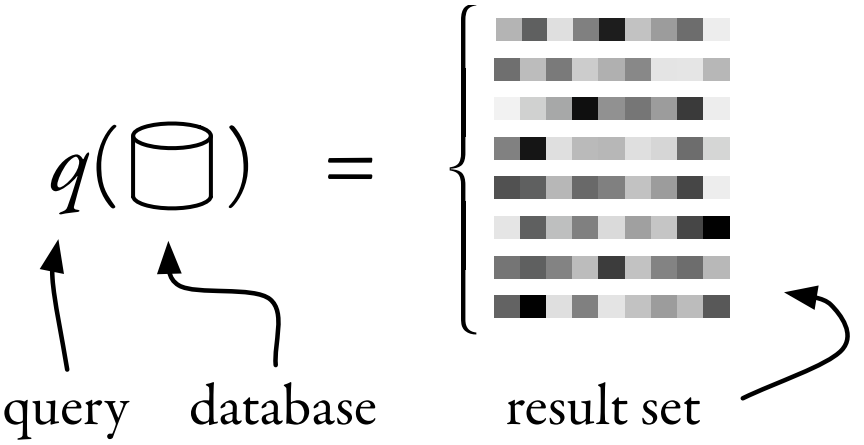
On decidable automata on data words and data trees in relation to satisfiability of LTL and XPath.

Diego Figueira

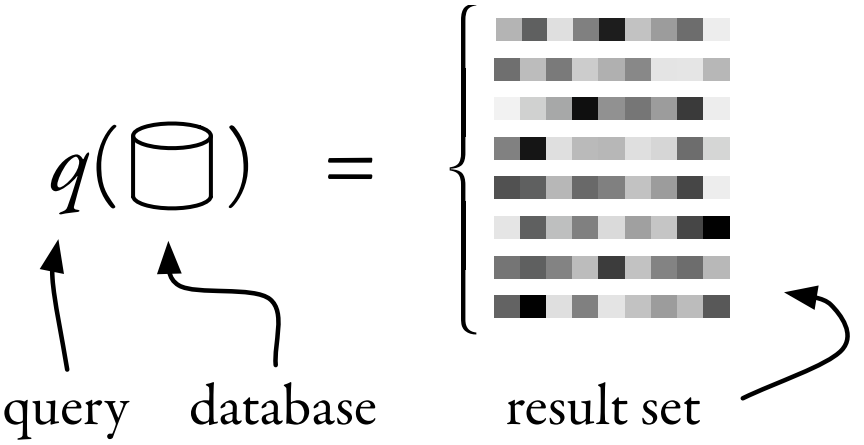
verification of databases



verification of databases

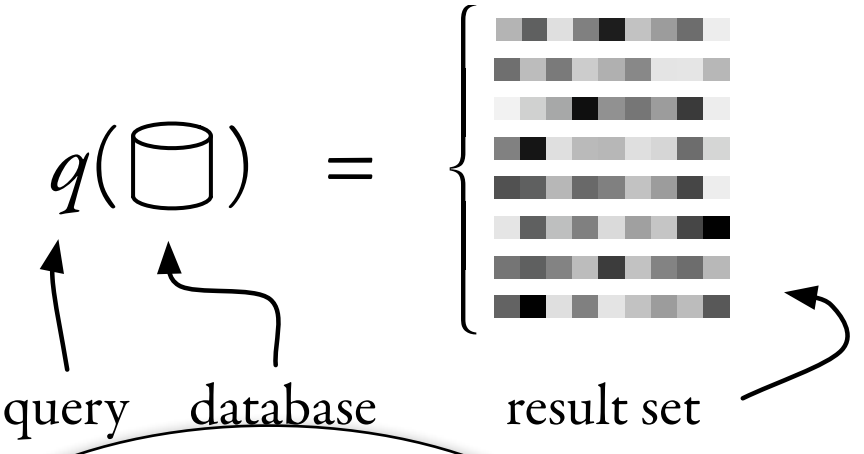


verification of databases



static analysis

verification of databases



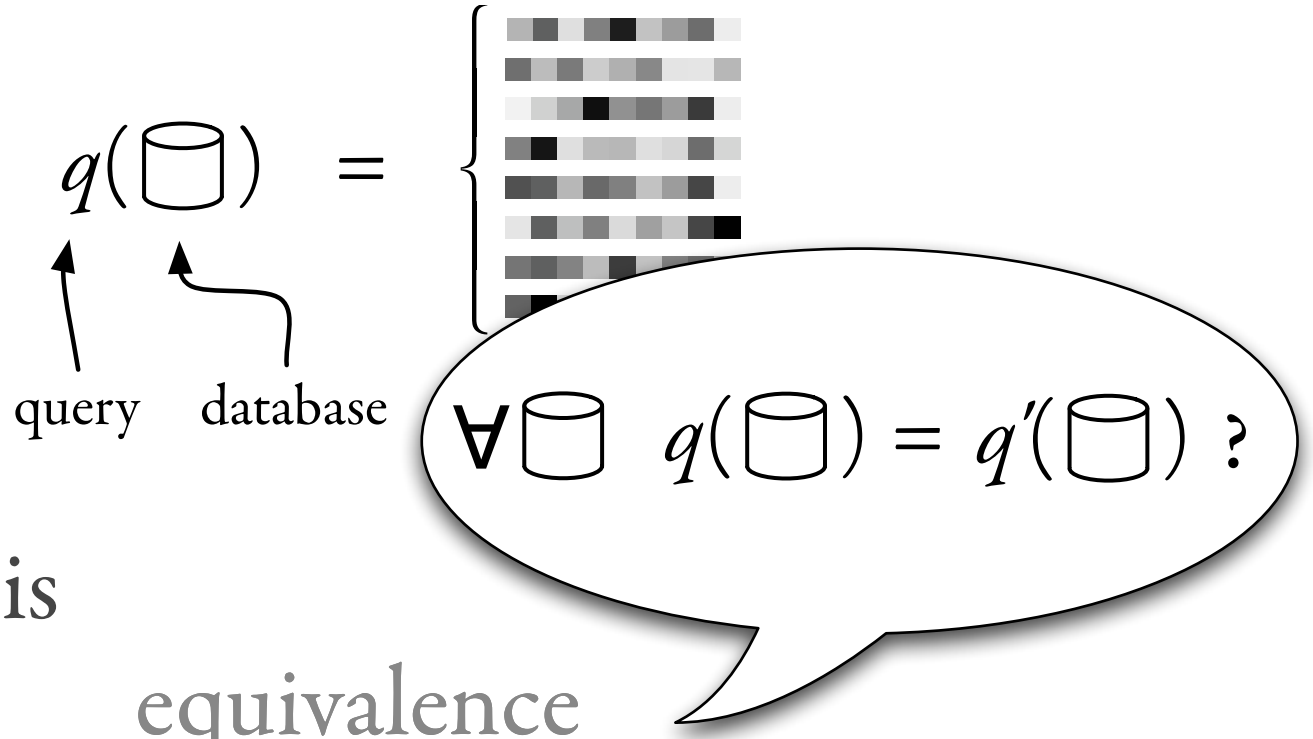
static anal

$\exists$ $q(\ ) \neq \varnothing$?

satisfiability

does *q* express a property?

verification of databases



static analysis

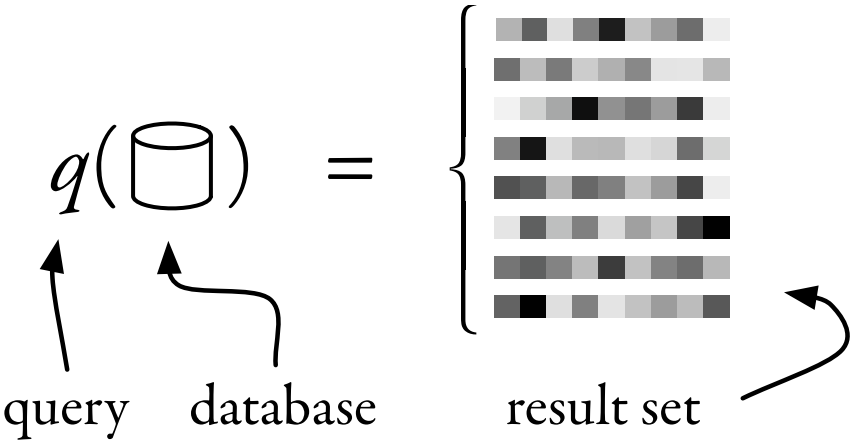
equivalence

can we use q' instead of q?

satisfiability

does q express a property?

verification of databases



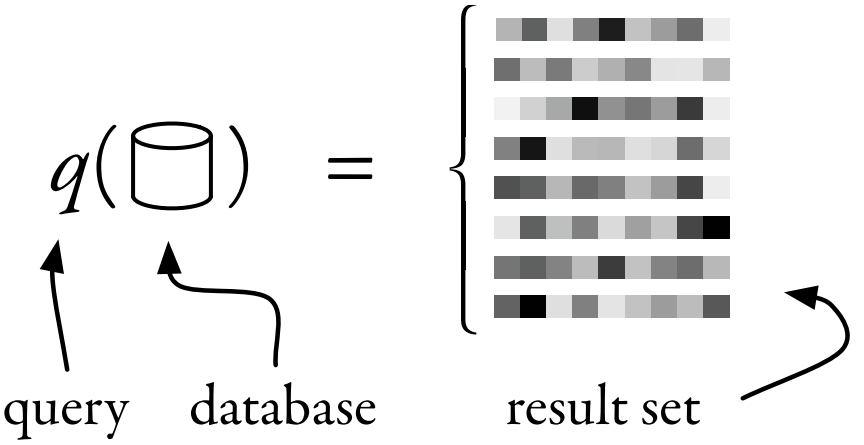
static analy

$\forall$ $q(\,) \subseteq q'(\,)$?

satisfiability
does q express a property?

containment
are q and q' comparable queries?

verification of databases



static analysis

equivalence

can we use q' instead of q?

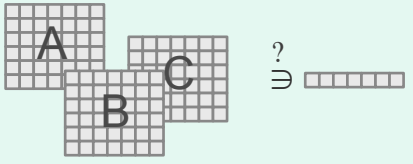
satisfiability

does q express a property?

containment

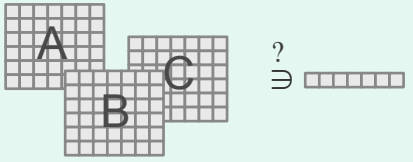
are q and q' comparable queries?

databases with structure

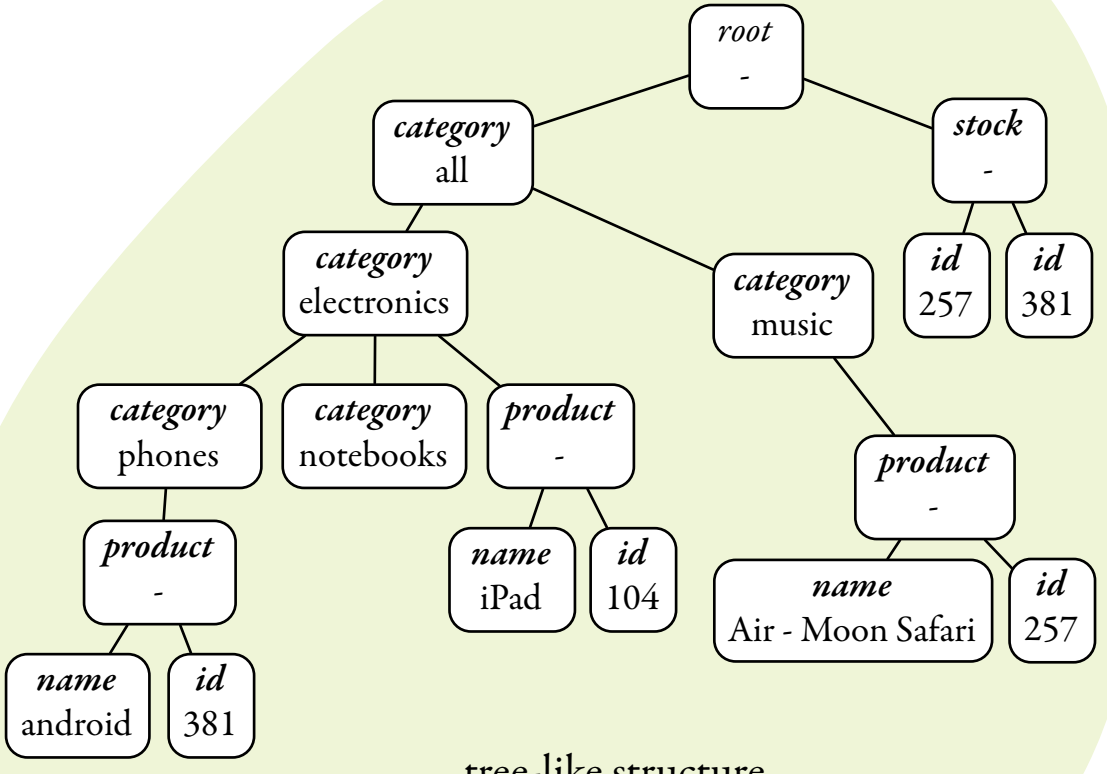


relational databases

databases with structure

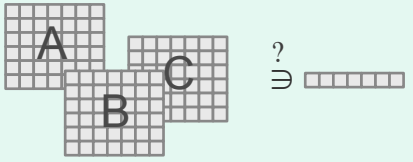


relational databases

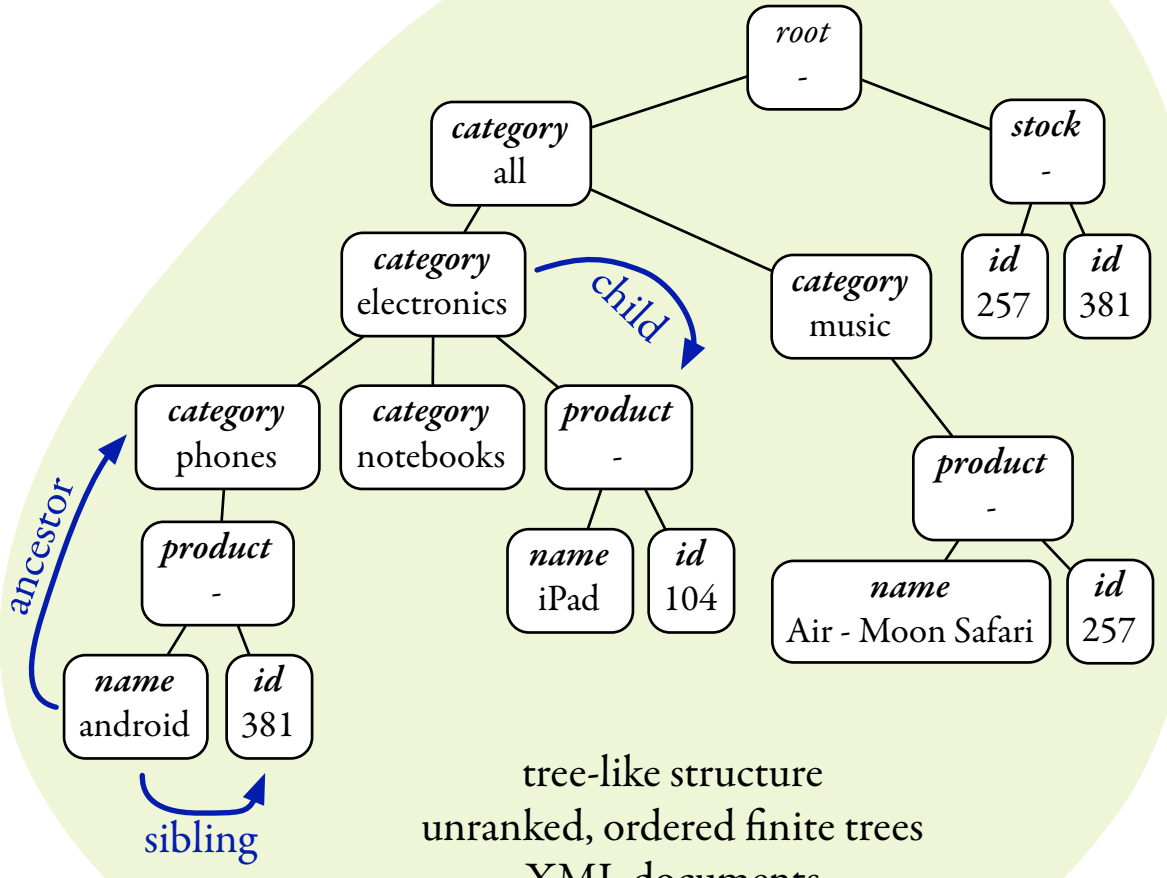


tree-like structure
unranked, ordered finite trees
XML documents
“data tree”

databases with structure

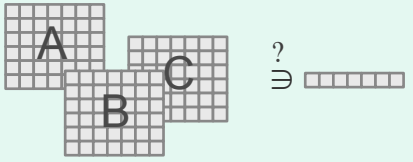


relational databases

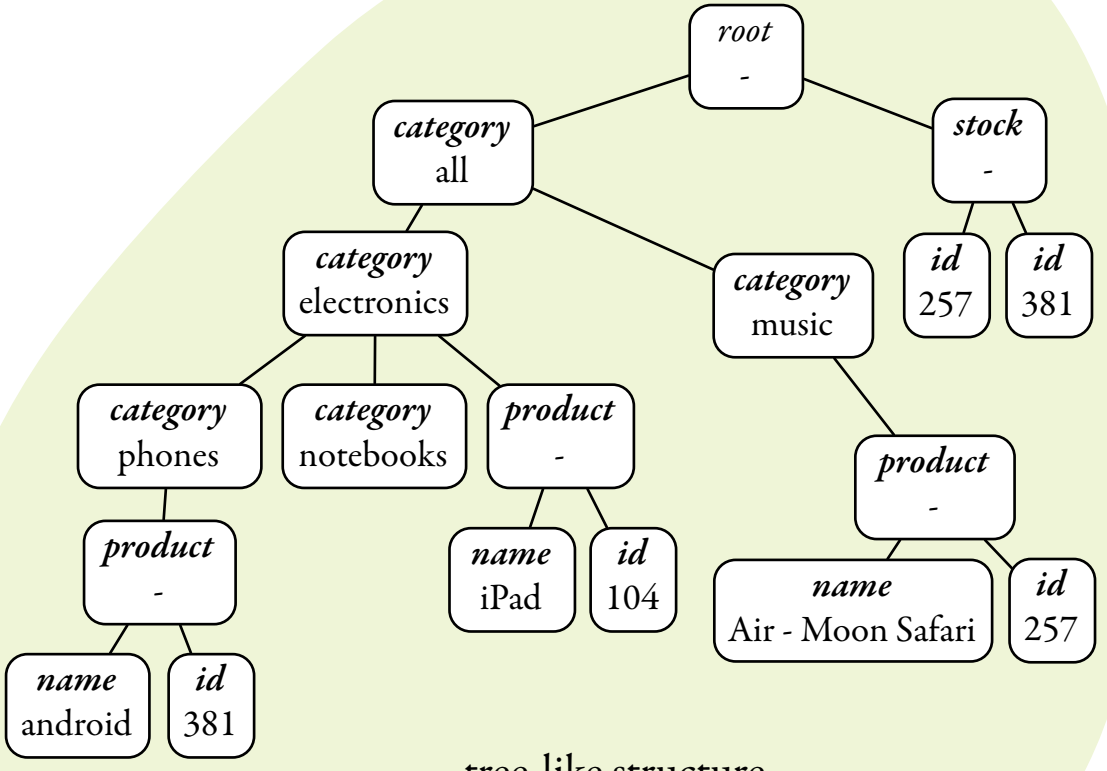


tree-like structure
unranked, ordered finite trees
XML documents
“data tree”

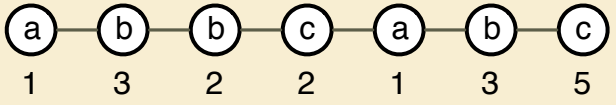
databases with structure



relational databases

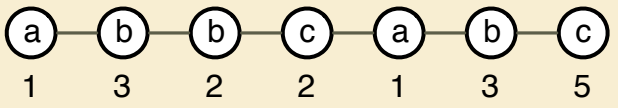


tree-like structure
unranked, ordered finite trees
XML documents
“data tree”

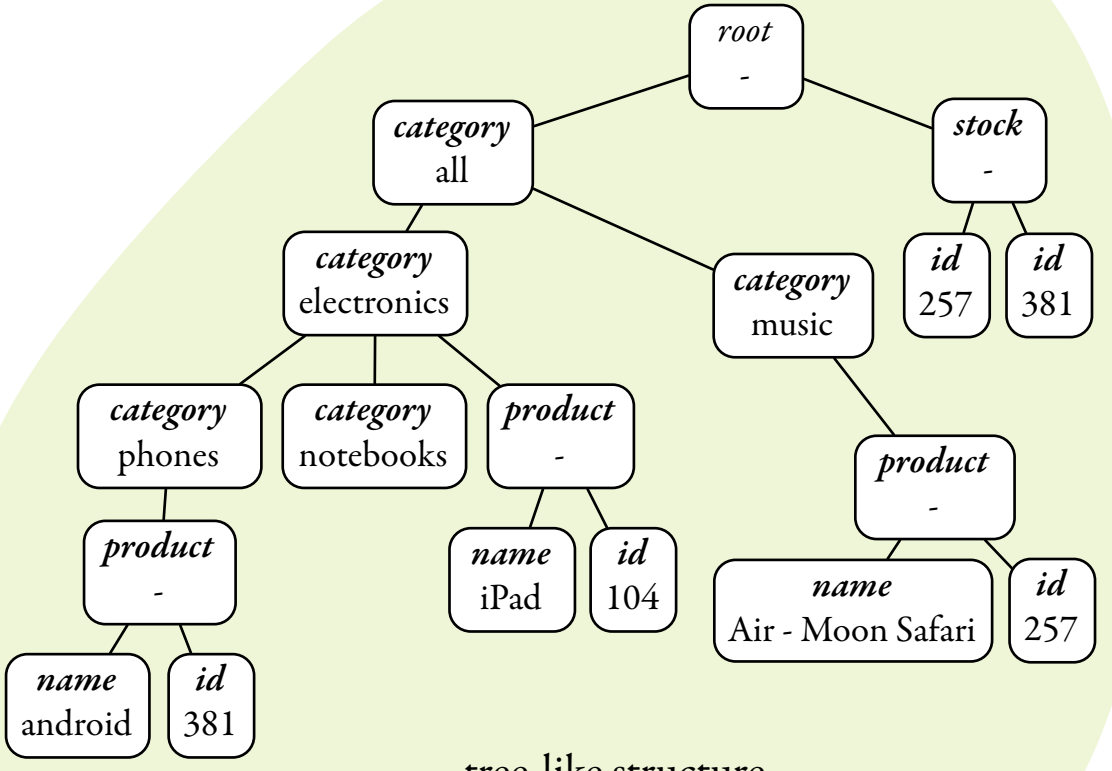


special case of a data tree
temporal databases
“data word”

databases with structure



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

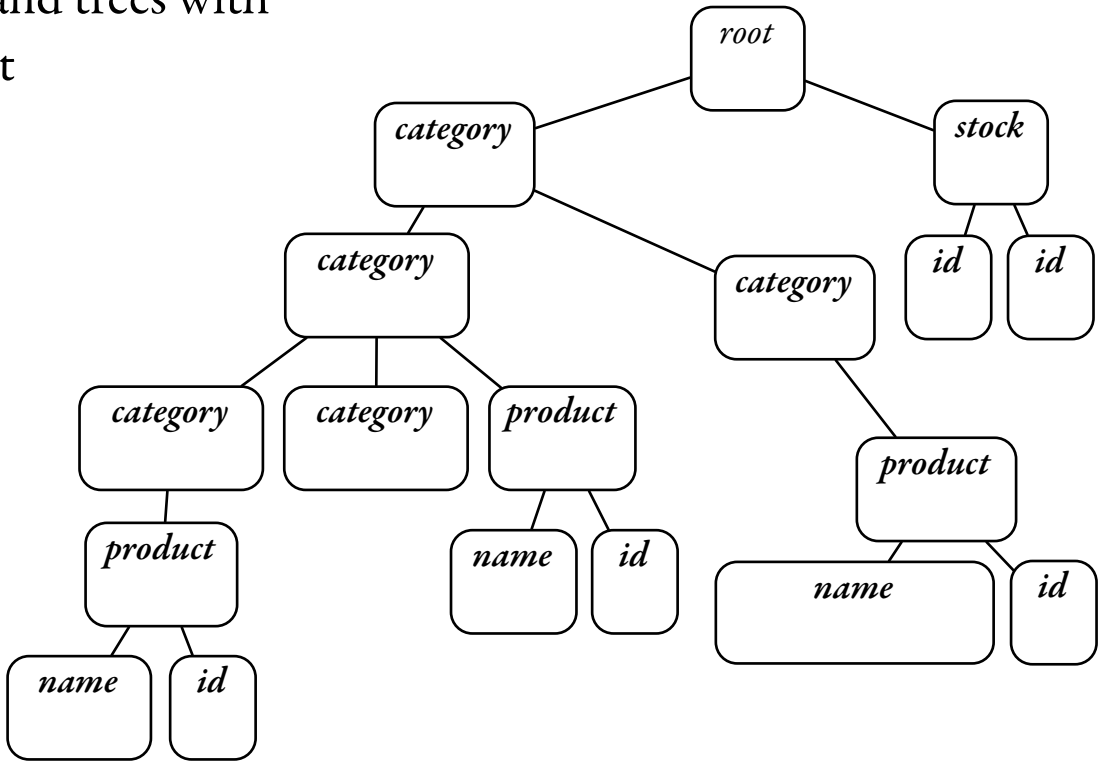


tree-like structure
unranked, ordered finite trees
XML documents
“data tree”

data values

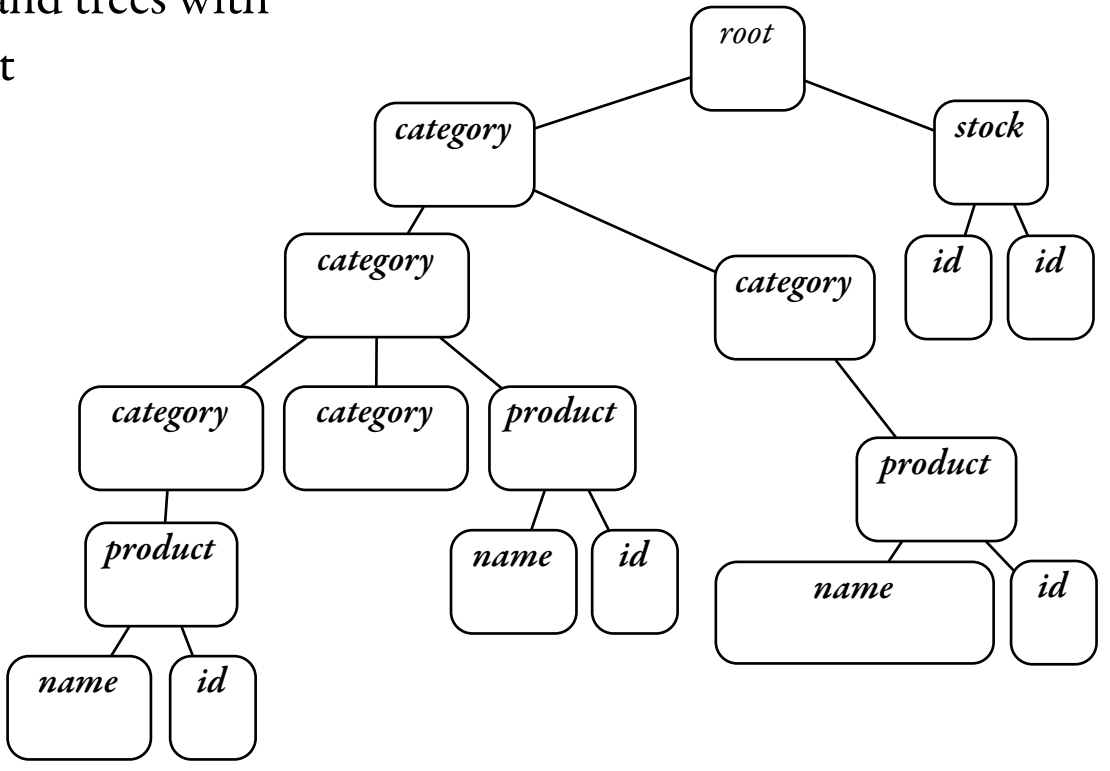
work on words and trees with

- finite alphabet
- structure



data values

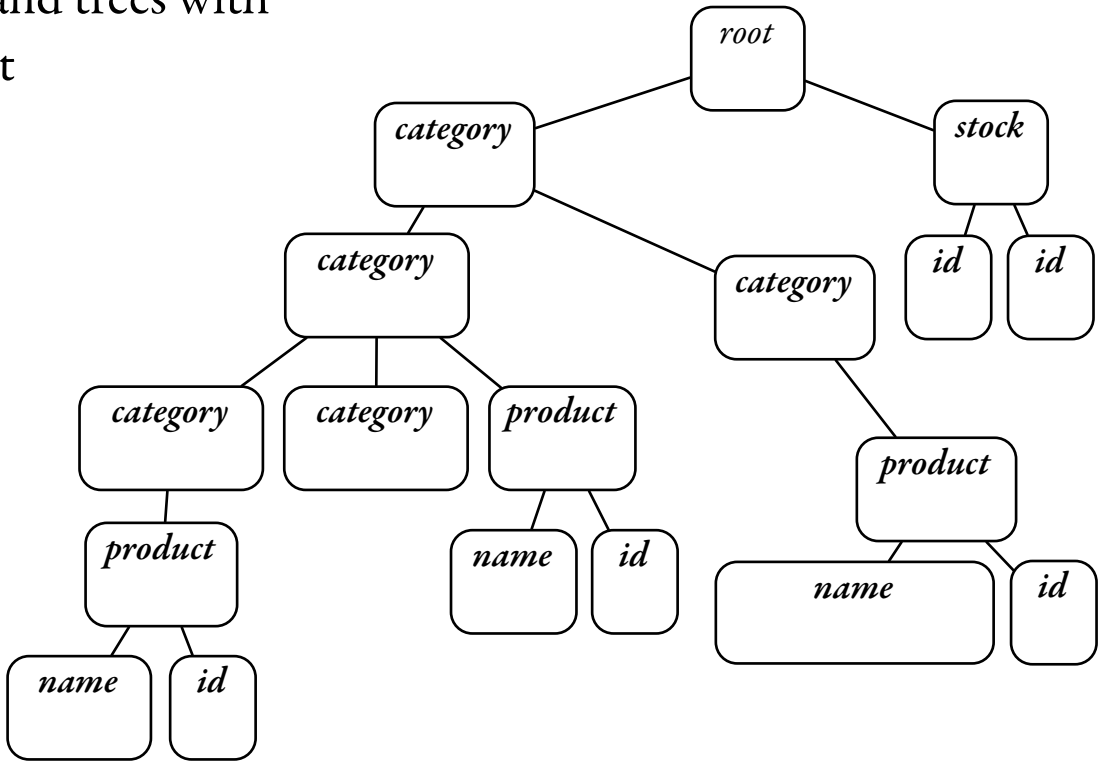
work on words and trees with
- finite alphabet
- structure



need to perform joins,
to test for equality of data values

data values

work on words and trees with
- finite alphabet
- structure

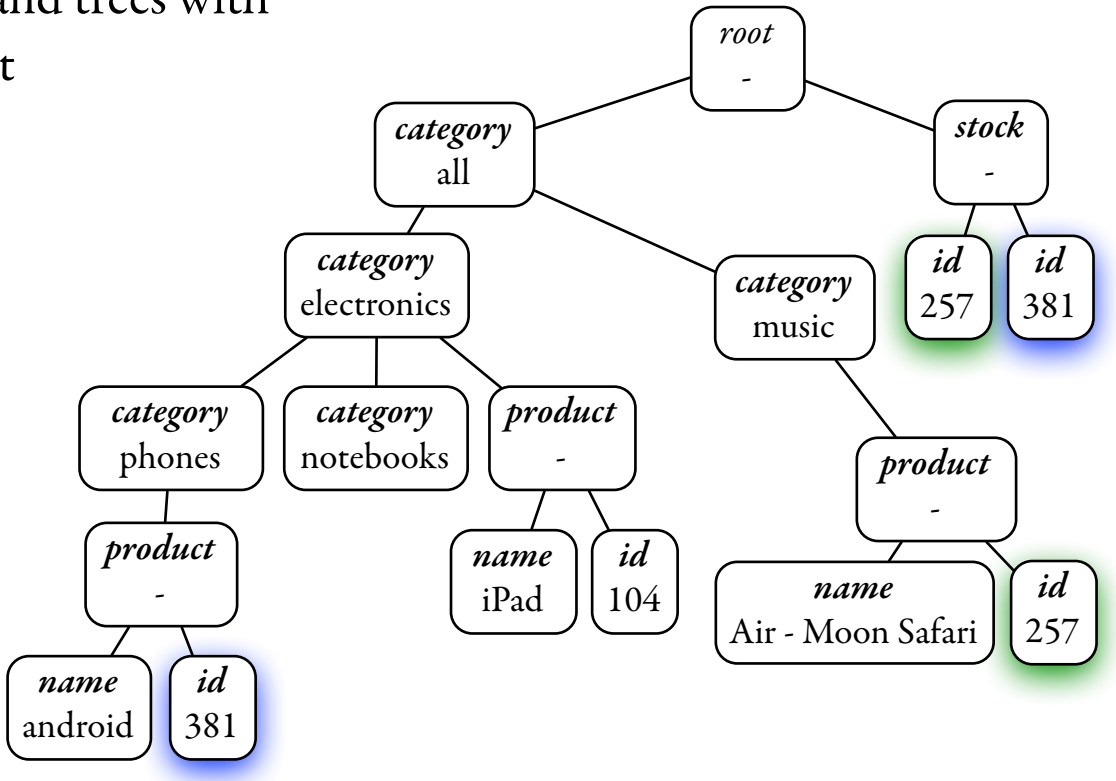


"return all products in stock"

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

FO^2
Bojańczyk, Muscholl, Schwentick, Segoufin, David, . . .

XPath
Benedikt, Fan, Geerts

patterns
conjunctive queries
David, Björklund, Martens, Schwentick

≈ [F, Hofman, Lasota, EXPRESS'10]

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

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, . . .

FO^2
Bojańczyk, Muscholl, Schwentick, Segoufin, David, . . .

XPath
Benedikt, Fan, Geerts

patterns
conjunctive queries
David, Björklund, Martens, Schwentick

≈ [F, Hofman, Lasota, EXPRESS'10]

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

LTL with registers
Demri, Lazić, Nowak

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

Hybrid logics
Description Logics
with concrete domains

objective

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

objective

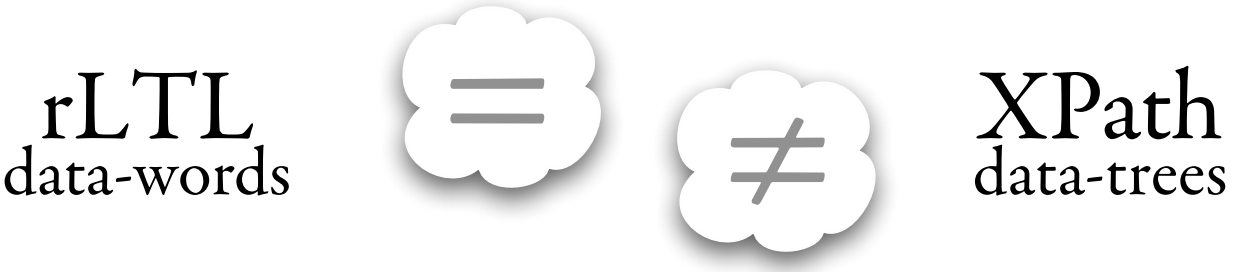
data-aware languages & automata for word- & tree- structured dbs

rLTL
data-words

XPath
data-trees

objective

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



objective

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

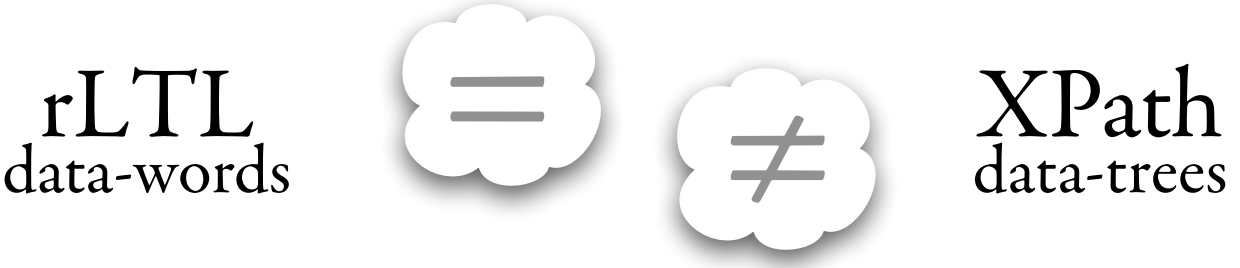
rLTL
data-words

XPath
data-trees

equivalence satisfiability inclusion

objective

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



equivalence satisfiability inclusion

objective

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

rLTL
data-words

=

≠

XPath
data-trees

equivalence satisfiability inclusion

rLTL(F,X)

register-LTL over data words

$\phi ::= a \mid \mathbf{F}\phi \mid \mathbf{X}\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \mid \downarrow\phi \mid \uparrow$

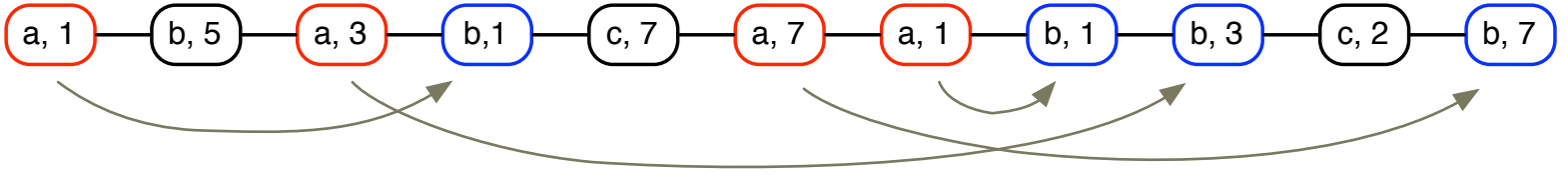
tests current datum against the stored

stores current datum

e.g.

"For every **a** there is a future **b** with the same value."

$$G\,(\ \neg \mathbf{a} \vee \downarrow F\,(\mathbf{b} \wedge \uparrow)\ \ )$$



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... [Demri / Lazić]

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!

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

[F, Segoufin, MFCS'09]

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

even for a very simple fragment

rLTL(U,X)

register-LTL over data words

$$\phi ::= a \mid \phi \mathbf{U} \phi \mid \mathbf{X}\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow$$

$\uparrow$ → tests current datum against the stored

$\downarrow \phi$ → stores current datum

e.g.

rLTL(U,X)

register-LTL over data words

$$\phi ::= a \mid \phi \mathbf{U} \phi \mid \mathbf{X}\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \mid \forall_{\leq} \phi \mid \exists_{\geq} \phi$$

$\exists_{\geq}\phi$: Exists a future data value s.t. ϕ.

$\forall_{\leq}\phi$: For all past data value, ϕ.

e.g.

rLTL(U,X)

register-LTL over data words

$$\phi ::= a \mid \phi \mathbf{U} \phi \mid \mathbf{X}\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \mid \forall_{\leq} \phi \mid \exists_{\geq} \phi$$

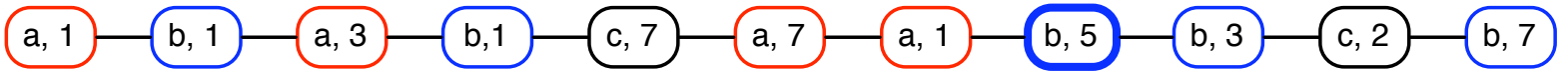
$\exists_{\geq}\phi$: Exists a future data value s.t. ϕ.

$\forall_{\leq}\phi$: For all past data value, ϕ.

e.g.

"There is a **b** with a data value different from any previous element."

$$F\,(\ \neg \mathbf{b}\ \vee\ \forall_{\leq}\,(\neg \uparrow)\ )$$



rLTL(U,X)

register-LTL over data words

$$\phi ::= a \mid \phi \mathbf{U} \phi \mid \mathbf{X}\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \mid \forall_{\leq} \phi \mid \exists_{\geq} \phi$$

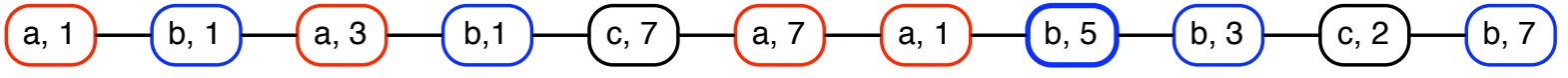
$\exists_{\geq} \phi$: Exists a future data value s.t. ϕ.

$\forall_{\leq} \phi$: For all past data value, ϕ.

e.g.

"There exists a data value that is contained in a **b** but not in an **a**."

$$\exists_{\geq} (\; F(\mathbf{b} \wedge \uparrow) \; \wedge \; \neg F(\mathbf{a} \wedge \uparrow) \;)$$



rLTL(U,X)

register-LTL over data words

$$\phi ::= a \mid \phi \mathbf{U} \phi \mid \mathbf{X}\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \mid \forall_{\leq} \phi \mid \exists_{\geq} \phi$$

$\exists_{\geq}\phi$: Exists a future data value s.t. ϕ.

$\forall_{\leq}\phi$: For all past data value, ϕ.

The satisfiability problem for **positive** rLTL(U,X) + $\forall_{\leq}$ + $\exists_{\geq}$ is **decidable**.

rLTL(U,X)

register-LTL over data words

$$\phi ::= a \mid \phi \mathbf{U} \phi \mid \mathbf{X}\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \mid \downarrow \phi \mid \uparrow \mid \forall_{\leq} \phi \mid \exists_{\geq} \phi$$

$\exists_{\geq}\phi$: Exists a future data value s.t. ϕ.

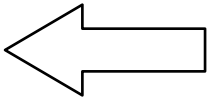
$\forall_{\leq}\phi$: For all past data value, ϕ.

The satisfiability problem for **positive** rLTL(U,X) + $\forall_{\leq}$ + $\exists_{\geq}$ is **decidable**.

The satisfiability problem for rLTL(U,X) + $\forall_{\leq}$ + $\exists_{\geq}$ is **undecidable**.

ARA

[Demri / Lazić]

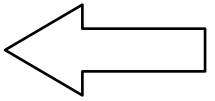


satisfiability of rLTL(U,X)

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

ARA

[Demri / Lazić]

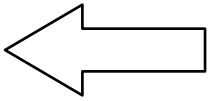
 satisfiability of rLTL(U,X)

- ☺ 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 complement preserving decidability

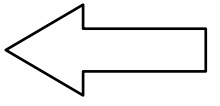
ARA
[Demri / Lazić]



satisfiability of rLTL(U,X)

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

ARA + guess + spread



satisfiability of positive rLTL(U,X) + $\forall\leq$ + $\exists\geq$

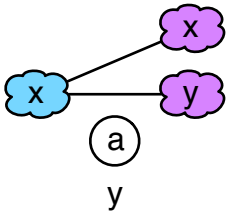
- ☺ Still decidable emptiness problem
- ☹ No longer closed under complementation
- ☠ Can't be closed under complement preserving decidability

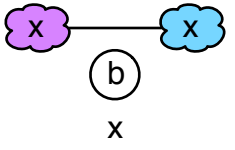
ARA
+guess +spread

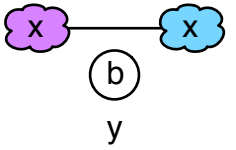
Alternating register automata
one-way
1 register

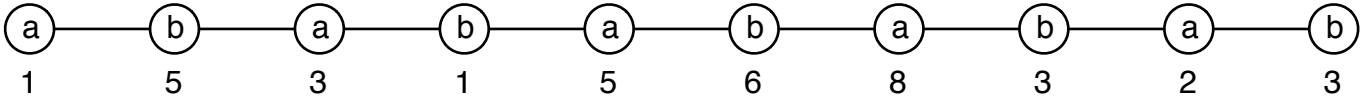
states:
initial state:
final states:

transitions







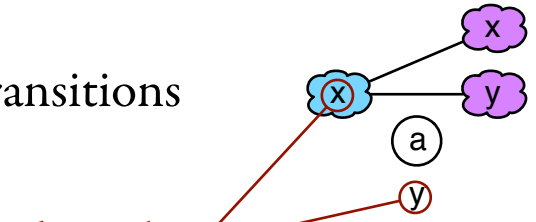


ARA
+guess +spread

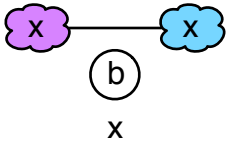
Alternating register automata
one-way
1 register

states:
initial state:
final states:

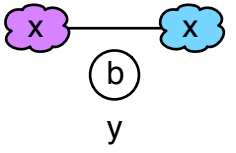
transitions



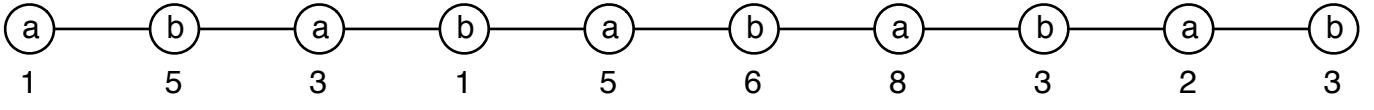
data values



x



y

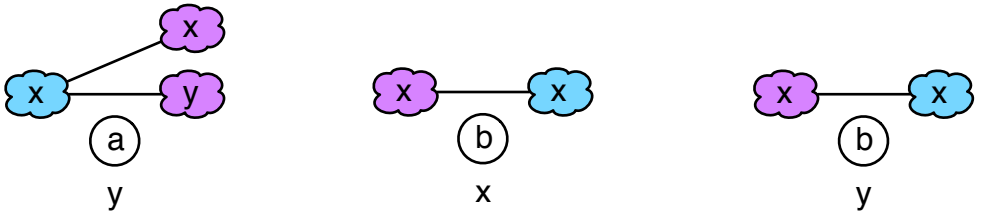


ARA
+guess +spread

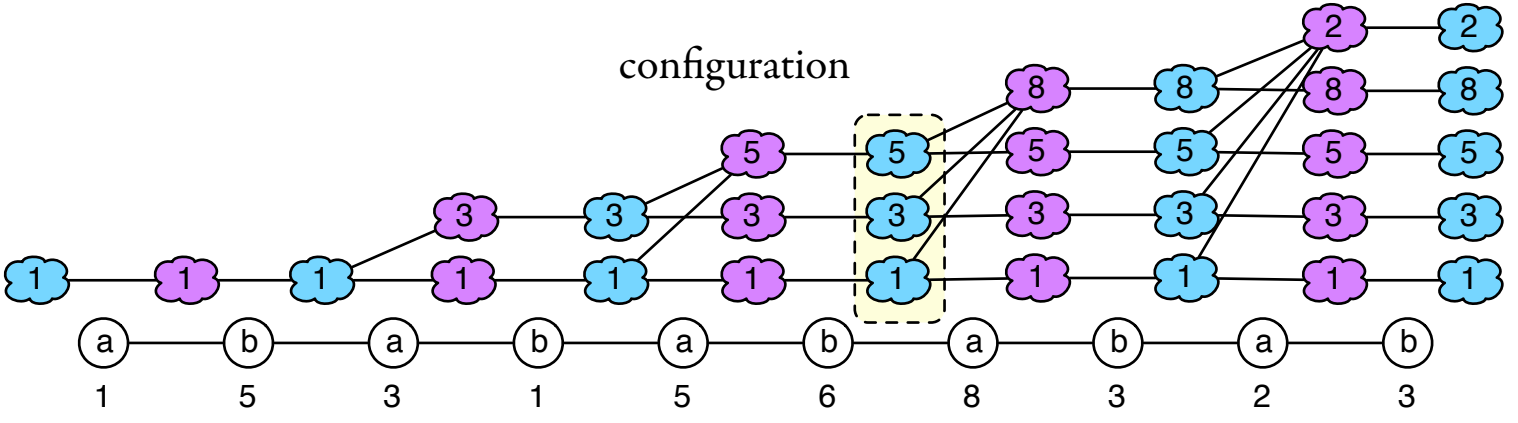
Alternating register automata
one-way
1 register

states:
initial state:
final states:

transitions



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



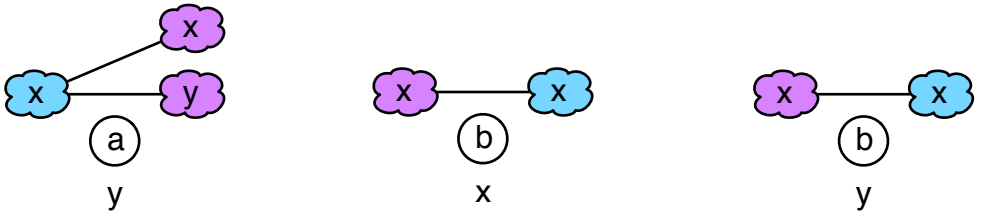
the string is $(ab)^*$, and all the a's have different data values

ARA
+guess +spread

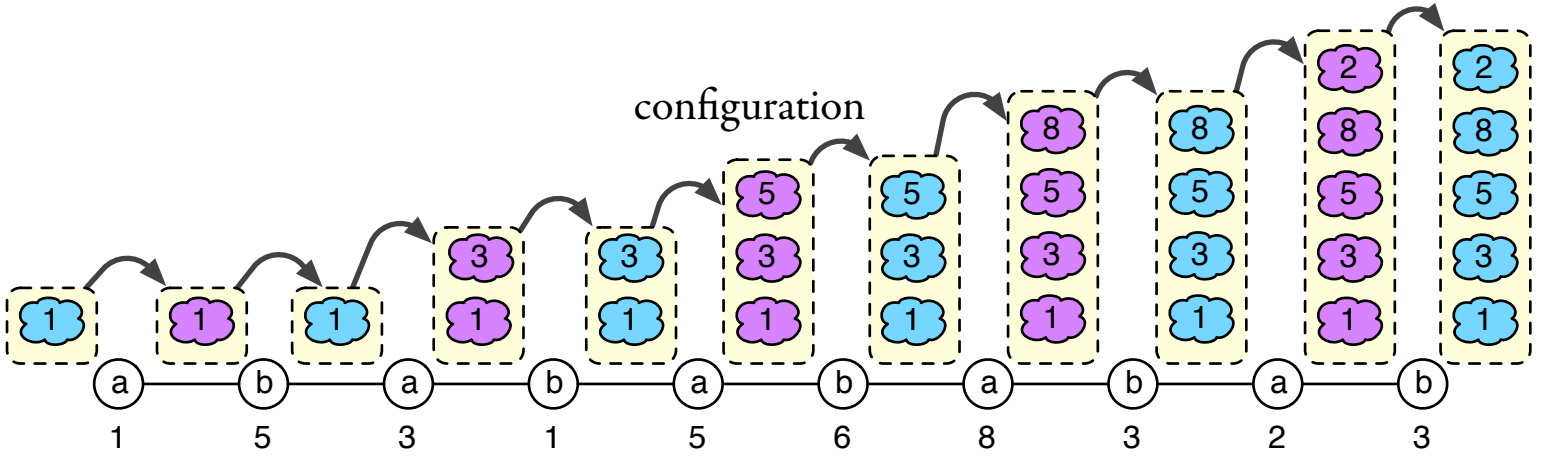
Alternating register automata
one-way
1 register

states:
initial state:
final states:

transitions



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



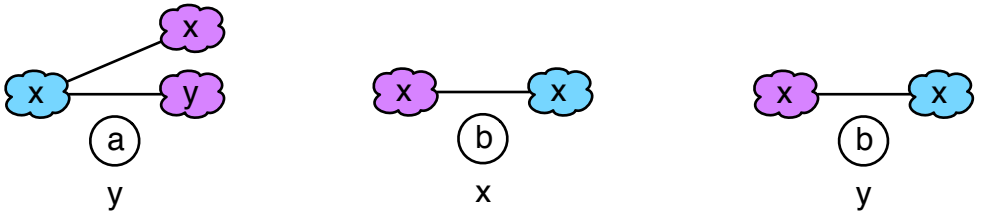
the string is $(ab)^*$, and all the a's have different data values

ARA
+guess +spread

Alternating register automata
one-way
1 register

states:
initial state:
final states:

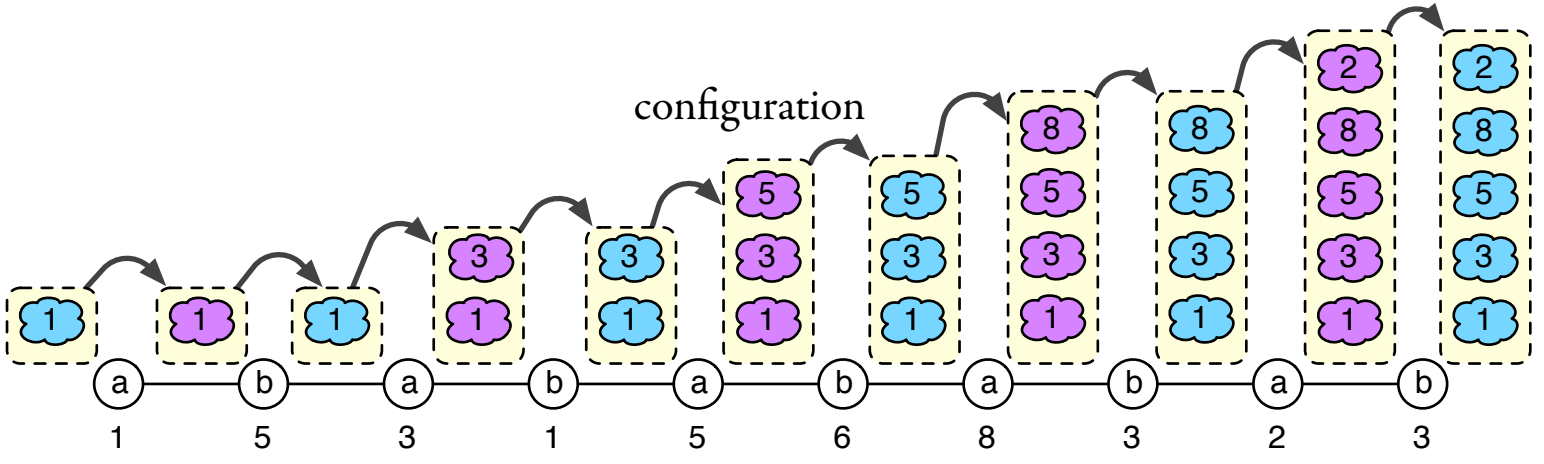
transitions



2 extensions

guess(q) Nondeterministically guess a data value and store it in the register.

spread(q, q') For every thread *(q,d)*, create a new one *(q',d)*.



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

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**

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

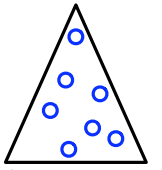
1. One that at every position create a thread that
   a. stores current data value, and
   b. advances until the last position using some state *q*
2. Other that
   a. guesses a b position of the word
   b. spreads all threads with *q* to threads with *p*

(state *p* checks that the stored data value is different from the current value)

XPath

node expressions

$\varphi, \psi ::= \mathbf{a} \mid \neg\varphi \mid \varphi \wedge \psi \mid \alpha{=}\beta \mid \alpha{\neq}\beta \mid \alpha? \qquad \mathbf{a} \in \mathcal{A}$

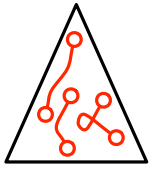


denote sets of nodes

path expressions

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

$o \in \{ \downarrow, \vdots, \rightarrow, \dashrightarrow, \uparrow, \vdots, \leftarrow, \dashleftarrow \}$



denote binary relations

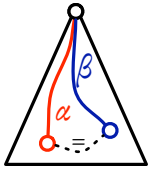
‘$\rightarrow$’ : *one-step*

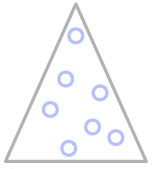
‘$\dashrightarrow$’ : *multistep* (transitive closure of ‘$\rightarrow$’)

XPath

node expressions

$\varphi, \psi ::= \mathbf{a} \mid \neg\varphi \mid \varphi \wedge \psi \mid \alpha{=}\beta \mid \alpha{\neq}\beta \mid \alpha? \qquad \mathbf{a} \in A$



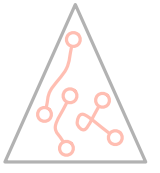


denote sets of nodes

path expressions

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

$o \in \{ \downarrow, \vdots, \rightarrow, \dashrightarrow, \uparrow, \vdots, \leftarrow, \dashleftarrow \}$



denote binary relations

‘$\rightarrow$’ : *one-step*

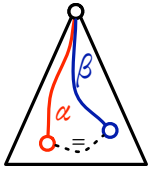
‘$\dashrightarrow$’ : *multistep* (transitive closure of ‘$\rightarrow$’)

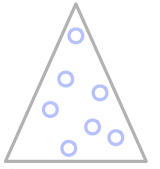
XPath

we note "XPath(=, $\rightarrow$, $\dashrightarrow$, $\uparrow$)"

node expressions

$\varphi, \psi ::= \mathbf{a} \mid \neg\varphi \mid \varphi \wedge \psi \mid \alpha{=}\beta \mid \alpha{\neq}\beta \mid \alpha? \qquad \mathbf{a} \in A$



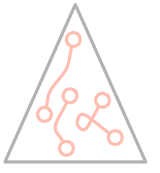


denote sets of nodes

path expressions

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

$o \in \{ \downarrow, \vdots, \rightarrow, \dashrightarrow, \uparrow, \vdots, \leftarrow, \dashleftarrow \}$

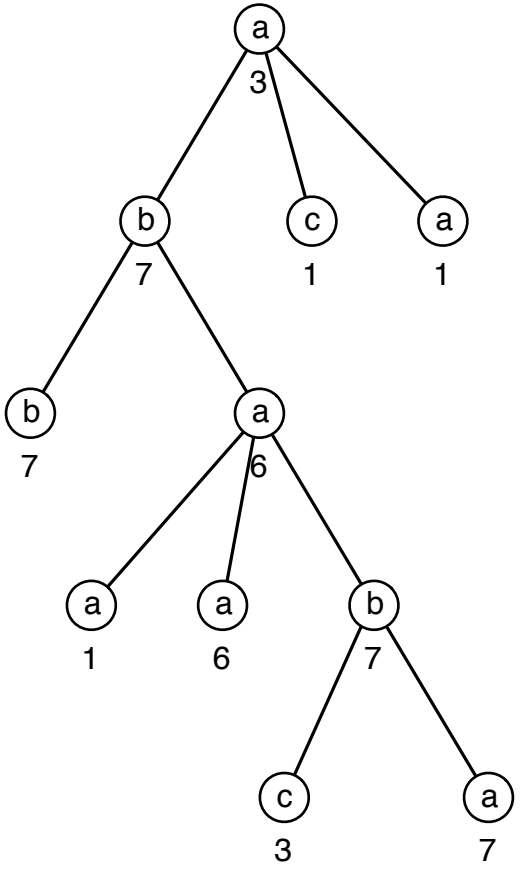


denote binary relations

'$\rightarrow$' : *one-step*

'$\dashrightarrow$' : *multistep* (transitive closure of '$\rightarrow$')

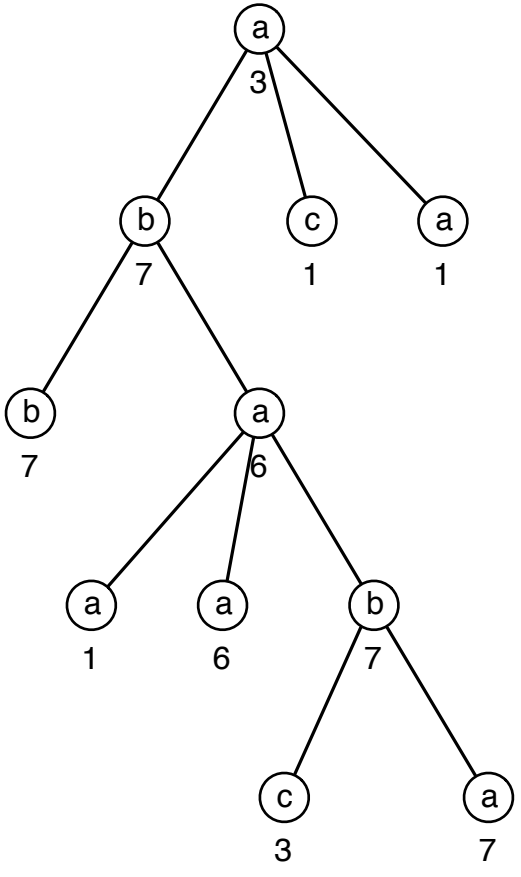
e.g.



⋮[a] = ⋮[b]

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

e.g.



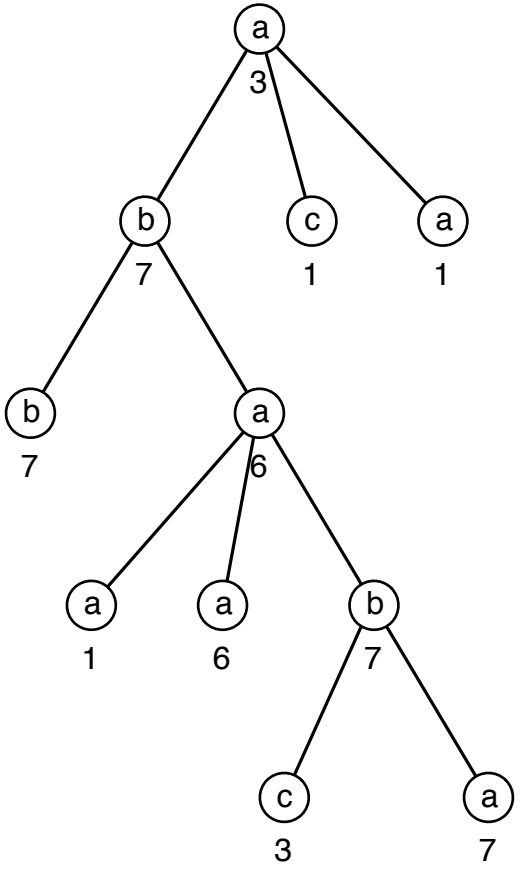
$\vdots [a] = \vdots [b]$

$\vdots [c] \neq \vdots [c]$

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

There are two ***c*** with different data values

e.g.



$\vdots[a] = \vdots[b]$

$\vdots[c] \neq \vdots[c]$

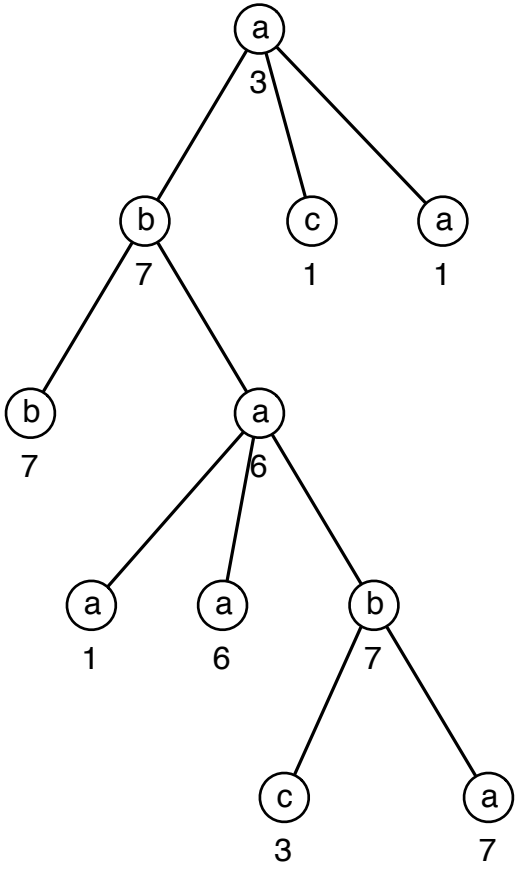
$\neg (\vdots[b] \neq \vdots[b])$

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

e.g.



$\vdots [a] = \vdots [b]$

$\vdots [c] \neq \vdots [c]$

$\neg (\vdots [b] \neq \vdots [b])$

$\neg (\vdots [b] = \vdots [c])$

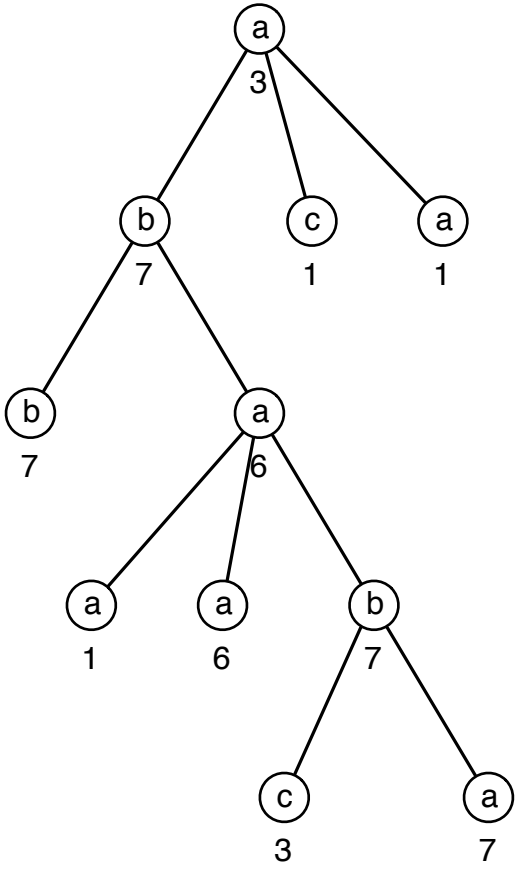
There are an ***a*** and a ***b*** with the same datum

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***

e.g.



$\vdots[a] = \vdots[b]$

$\vdots[c] \neq \vdots[c]$

$\neg (\vdots[b] \neq \vdots[b])$

$\neg (\vdots[b] = \vdots[c])$

$\neg (\vdots[\ \vdots[a] = \longrightarrow \cdots\blacktriangleright \vdots[a] \ ])$

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

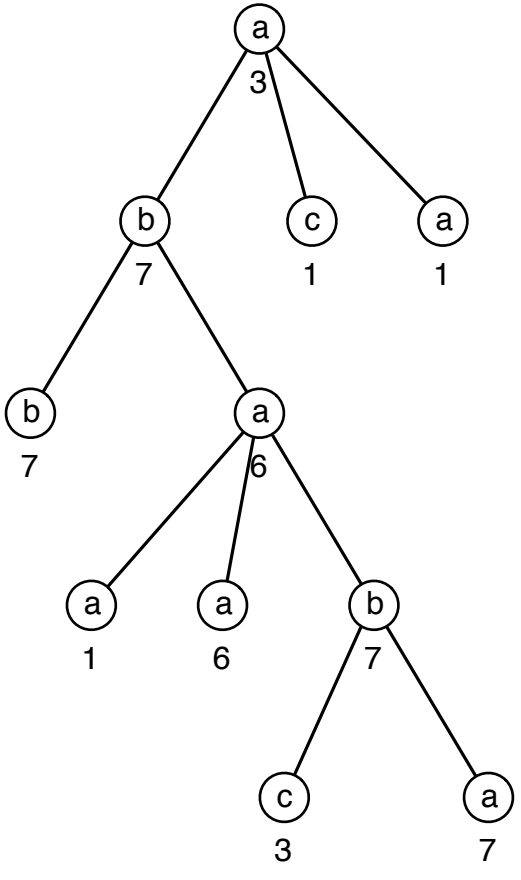
There are two ***c*** with different data values

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

e.g.



$\vdots[a] = \vdots[b]$

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

$\vdots[c] \neq \vdots[c]$

There are two ***c*** with different data values

$\neg (\vdots[b] \neq \vdots[b])$

All ***b*** have the same data value

$\neg (\vdots[b] = \vdots[c])$

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

$\neg (\vdots [\vdots[a] = \longrightarrow \cdots\blacktriangleright \vdots[a]])$

The ***c*** constitutes a primary key

$\neg (\vdots [a \wedge \neg (\varepsilon \neq \updownarrow[b])])$

Every ***a*** has an ancestor ***b*** with different value

satisfiability of XPath

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

satisfiability of XPath

Satisfiability of $\mathrm{XPath}(=, \downarrow, \vdots, \rightarrow, \cdots\blacktriangleright, \uparrow, \vdots, \leftarrow, \blacktriangleleft\cdots)$ is undecidable.

satisfiability of XPath

Satisfiability of $\mathrm{XPath}(=, \downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \uparrow, \uparrow^*, \leftarrow, \leftarrow^*)$ is undecidable.

satisfiability of XPath

Our contribution:

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$, $\rightarrow$, $\rightarrow^*$, $\uparrow$, $\uparrow^*$, $\leftarrow$, $\leftarrow^*$) is undecidable. [F, Segoufin, MFCS'09]

satisfiability of XPath

Our contribution:

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$, $\rightarrow$, $\rightarrow^*$, $\uparrow$, $\uparrow^*$, $\leftarrow$, $\leftarrow^*$) is undecidable. [F, Segoufin, MFCS'09]

satisfiability of XPath

Our contribution:

Satisfiability of $\mathrm{XPath}(=, \downarrow, \vdots, \rightarrow, \dashrightarrow, \uparrow, \vdots, \leftarrow, \dashleftarrow)$ is undecidable. [F, Segoufin, MFCS'09]

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

satisfiability of XPath

Our contribution:

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$, $\rightarrow$, $\rightarrow^*$, $\uparrow$, $\uparrow^*$, $\leftarrow$, $\leftarrow^*$) is undecidable. [F, Segoufin, MFCS'09]

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$) is ExpTime-complete. [F, PODS'09]

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$, $\rightarrow$, $\rightarrow^*$) is decidable. [F, ICDT'10]

satisfiability of XPath

Our contribution:

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$, $\rightarrow$, $\rightarrow^*$, $\uparrow$, $\uparrow^*$, $\leftarrow$, $\leftarrow^*$) is undecidable. [F, Segoufin, MFCS'09]

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$) is ExpTime-complete. [F, PODS'09]

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$, $\rightarrow$, $\rightarrow^*$) is decidable. [F, ICDT'10]

Satisfiability of XPath(=, $\downarrow$, $\downarrow^*$, $\uparrow$, $\uparrow^*$) is decidable. [F, Segoufin, STACS'11]

satisfiability of XPath

Our contribution:

Satisfiability of $\mathrm{XPath}(=, \downarrow, \vdots, \rightarrow, \cdots\!\blacktriangleright, \uparrow, \vdots, \leftarrow, \cdots\!\blacktriangleleft)$ is undecidable. [F, Segoufin, MFCS'09]

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

Satisfiability of $\mathrm{XPath}(=, \downarrow, \vdots, \rightarrow, \cdots\!\blacktriangleright)$ is decidable. [F, ICDT'10]

Satisfiability of $\mathrm{XPath}(=, \downarrow, \vdots, \uparrow, \vdots)$ is decidable. [F, Segoufin, STACS'11]

forward-XPath

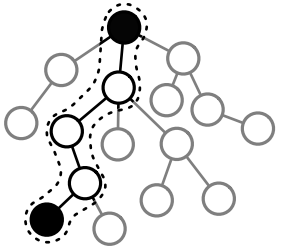
Satisfiability of $\mathrm{XPath}(=, \downarrow, \vdots \rightarrow, \cdots\blacktriangleright)$

forward-XPath

Satisfiability of ε-XPath$(=, \downarrow, \vdots \rightarrow, \cdots\blacktriangleright)$ is decidable.

[Lazić, Jurdziński, '07]

only tests: $\varepsilon = \alpha$

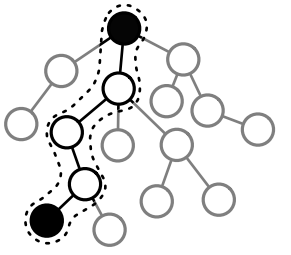


forward-XPath

Satisfiability of ε-XPath$(=, \downarrow, \vdots \rightarrow, \cdots\blacktriangleright)$ is decidable.

[Lazić, Jurdziński, '07]

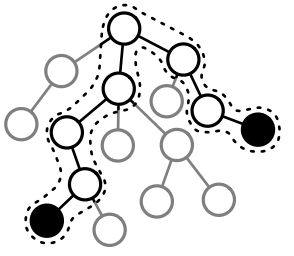
only tests: $\varepsilon = \alpha$



We show

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

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

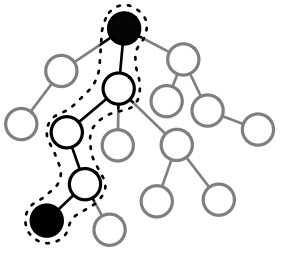


forward-XPath

Satisfiability of ε-XPath$(=, \downarrow, \vdots \rightarrow, \cdots\blacktriangleright)$ is decidable.

[Lazić, Jurdziński, '07]

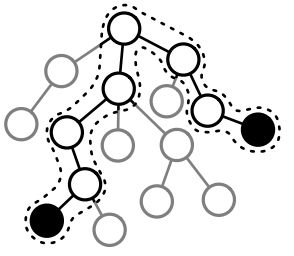
only tests: $\varepsilon = \alpha$



We show

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

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



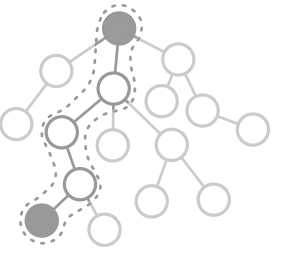
forward XPath ⟹ ATRA + guess + spread

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

forward-XPath

Satisfiability of ε-XPath$(=, \downarrow, \vdots \rightarrow, \cdots\blacktriangleright)$ is decidable.
[Lazić, Jurdziński, '07]

only tests: $\varepsilon = \alpha$



We show

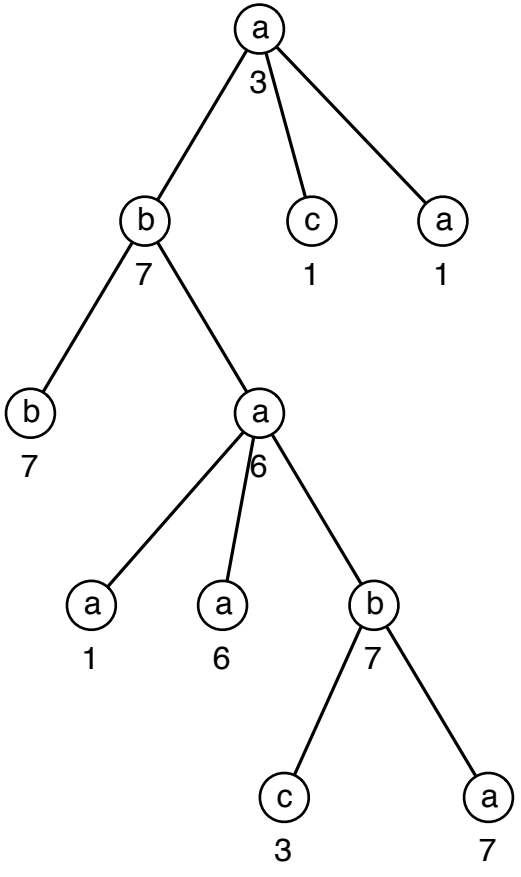
Satisfiability of XPath$(=, \downarrow, \vdots \rightarrow, \cdots\blacktriangleright)$ is decidable.
[F, ICDT'10] lower bound: non primitive recursive

forward XPath ⇒ ATRA + guess + spread

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

difficulty: ATRA$_{+\text{guess}+\text{spread}}$ are not closed under complementation

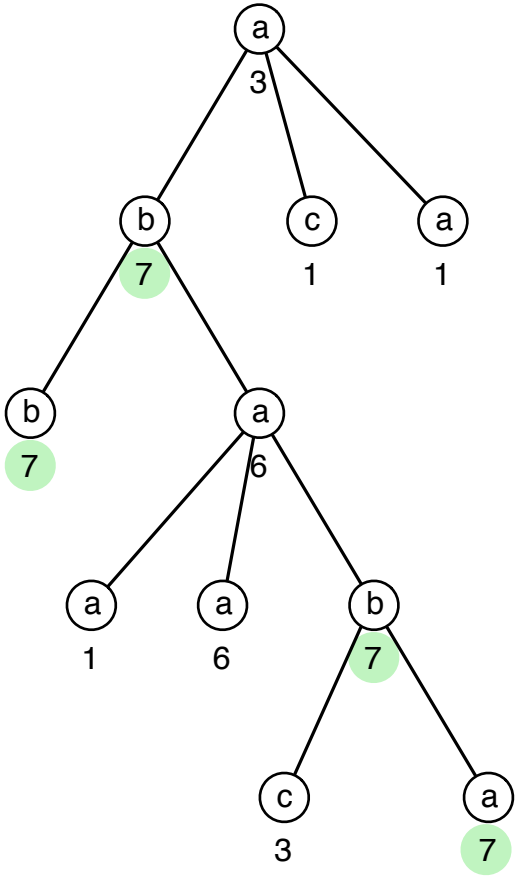
how do we code...?



⋮[a] = ⋮[b]

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

how do we code...?

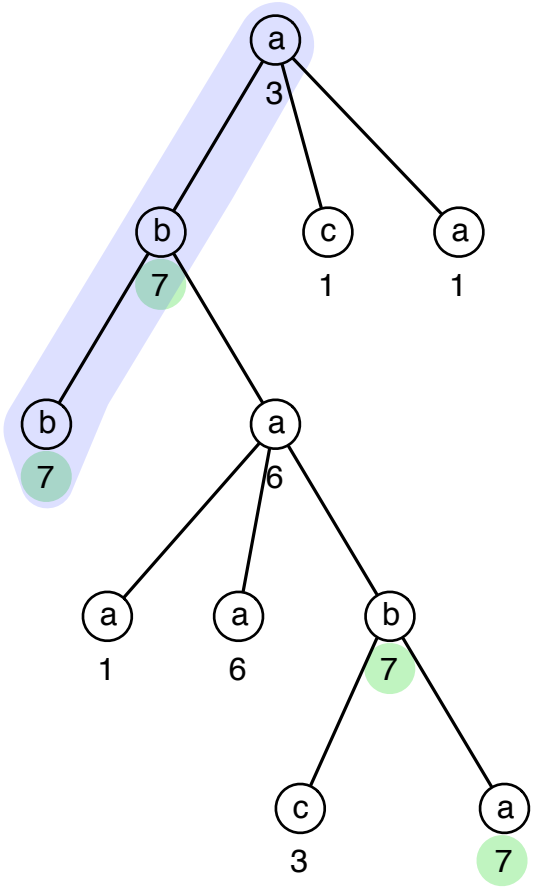


⋮[a] = ⋮[b]

Guess the data value 7.

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

how do we code...?



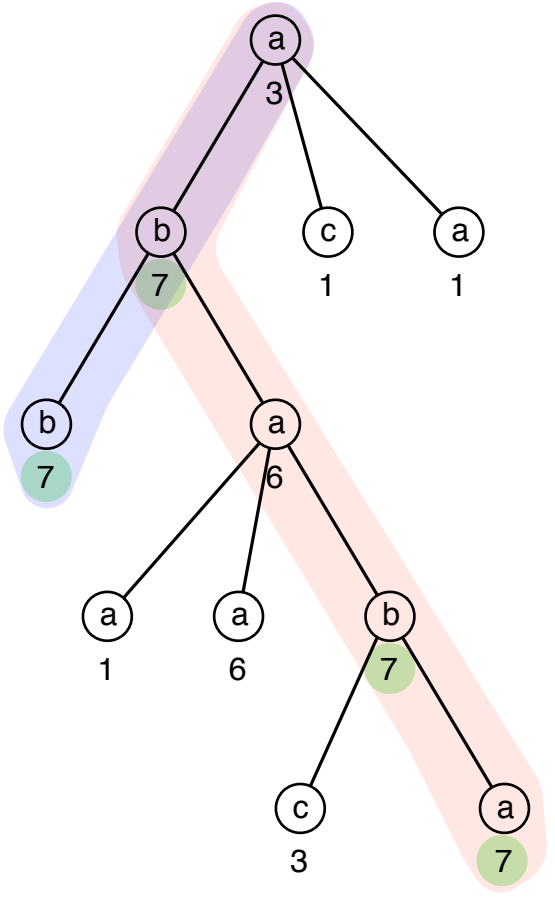
⋮[a] = ⋮[b]

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

Guess the data value 7.

Check that it can be accessed with " ⋮[b] ".

how do we code...?



$\vdots[a] = \vdots[b]$

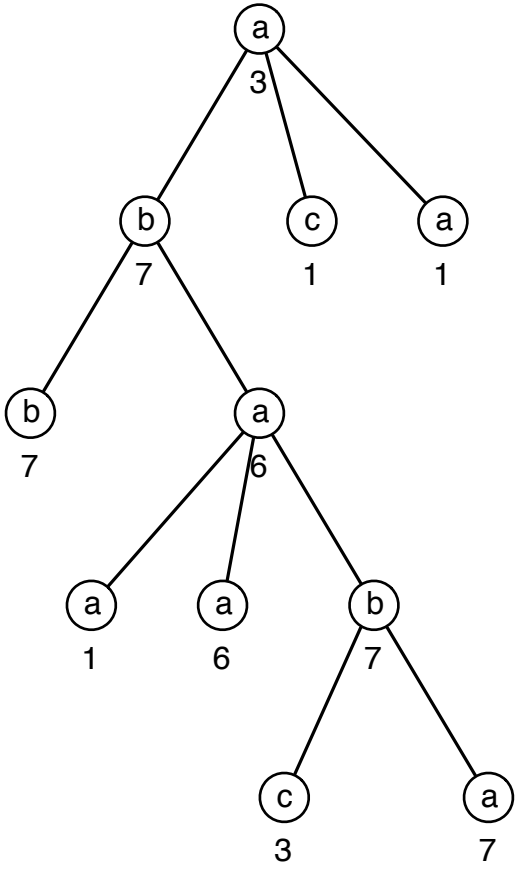
There are an ***a*** and a ***b*** with the same datum

Guess the data value 7.

Check that it can be accessed with " $\vdots[b]$ ".

Check that it can be accessed with " $\vdots[a]$ ".

how do we code...?



$\vdots\!\downarrow [a] = \vdots\!\downarrow [b]$

$\vdots\!\downarrow [c] \neq \vdots\!\downarrow [c]$

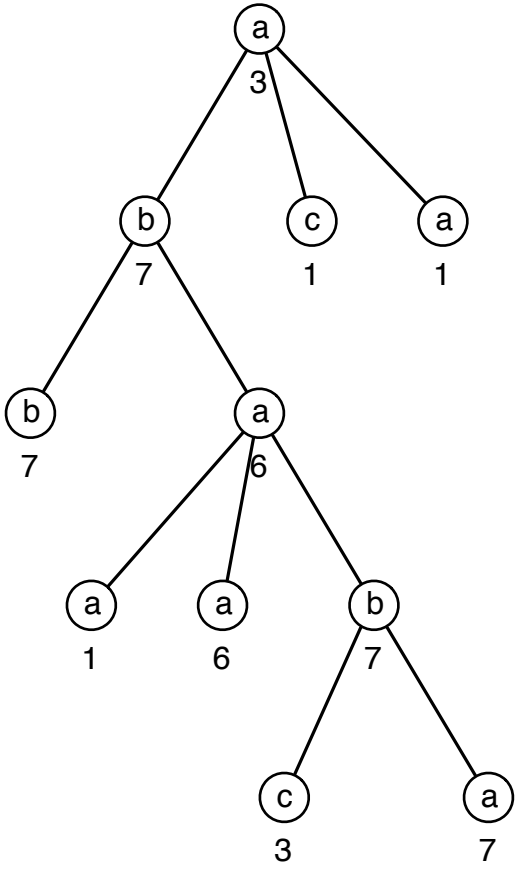
$\neg (\vdots\!\downarrow [b] \neq \vdots\!\downarrow [b])$

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...?



$\vdots [a] = \vdots [b]$ ✔

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

$\vdots [c] \neq \vdots [c]$ ✔

There are two ***c*** with different data values

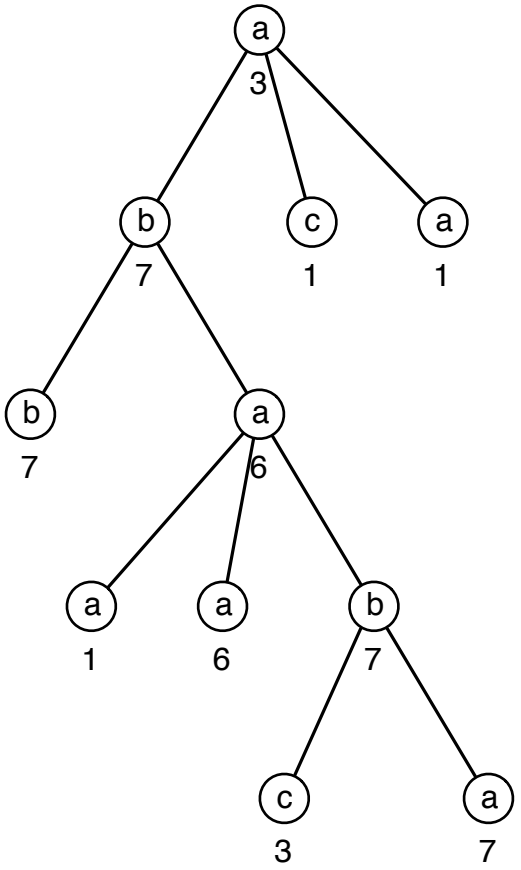
$\neg (\vdots [b] \neq \vdots [b])$ ✔

All ***b*** have the same data value

$\neg (\vdots [b] = \vdots [c])$ ✘

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

how do we code...?



$\vdots[a] = \vdots[b]$ ✔

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

$\vdots[c] \neq \vdots[c]$ ✔

There are two ***c*** with different data values

$\neg(\vdots[b] \neq \vdots[b])$ ✔

All ***b*** have the same data value

$\neg(\vdots[b] = \vdots[c])$ ✘

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

build automaton **A** such that

if $\mathbf{T} \models \varphi$, then **A** accepts **T**

if **A** accepts **T**, then there is $\mathbf{T}' = f(\mathbf{A},\mathbf{T})$ such that $\mathbf{T}' \models \varphi$

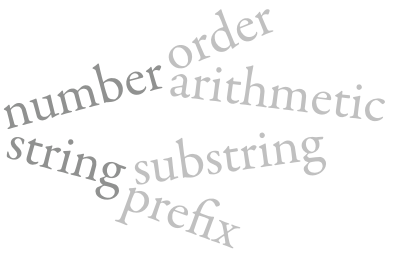
future work

XPath: inclusion&equivalence of *path expressions*

future work

XPath: inclusion&equivalence of *path expressions*

More domain specific operations



future work

XPath: inclusion&equivalence of *path expressions*

More domain specific operations number order arithmetic string substring prefix

Quest for an expressive formalism logic automaton
with *low* complexity

future work

XPath: inclusion&equivalence of *path expressions*

More domain specific operations number order arithmetic string substring prefix

Quest for an expressive formalism logic automaton
with *low* complexity

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

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

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(=, $\downarrow$, $\vdots$) 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(=, $\downarrow$, $\vdots$) is ExpTime-complete

DD automaton is 2ExpTime

3. Forward

XPath(=, $\downarrow$, $\vdots$,$\rightarrow$,$\cdots\blacktriangleright$) is decidable, non primitive recursive

ATRA(guess, spread) is decidable

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(=, $\downarrow$, $\downarrow^*$) is ExpTime-complete

DD automaton is 2ExpTime

3. Forward

XPath(=, $\downarrow$, $\downarrow^*$,$\rightarrow$,$\rightarrow^*$) is decidable,
non primitive recursive

ATRA(guess, spread) is decidable

4. Vertical

XPath(=, $\downarrow$, $\downarrow^*$, $\uparrow$, $\uparrow^*$) is decidable,
non primitive recursive

BUDA is decidable

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(=, $\downarrow$, $\downarrow^+$) is ExpTime-complete

DD automaton is 2ExpTime

3. Forward

XPath(=, $\downarrow$, $\downarrow^+$,$\rightarrow$,$\rightarrow^+$) is decidable, non primitive recursive

ATRA(guess, spread) is decidable

4. Vertical

XPath(=, $\downarrow$, $\downarrow^+$, $\uparrow$, $\uparrow^+$) is decidable, non primitive recursive

BUDA is decidable

Tomorrow 16h45