Two variable logics in the presence of an equivalence relation.

Bonn

September 2006

Thomas Schwentick

Joint work with

Mikołaj Bojańczyk, Claire David, Anca Muscholl, Luc Segoufin

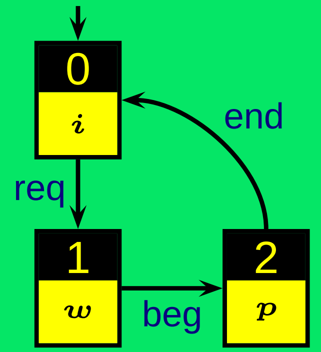
Toy examples from Computer Science

A printer and a process

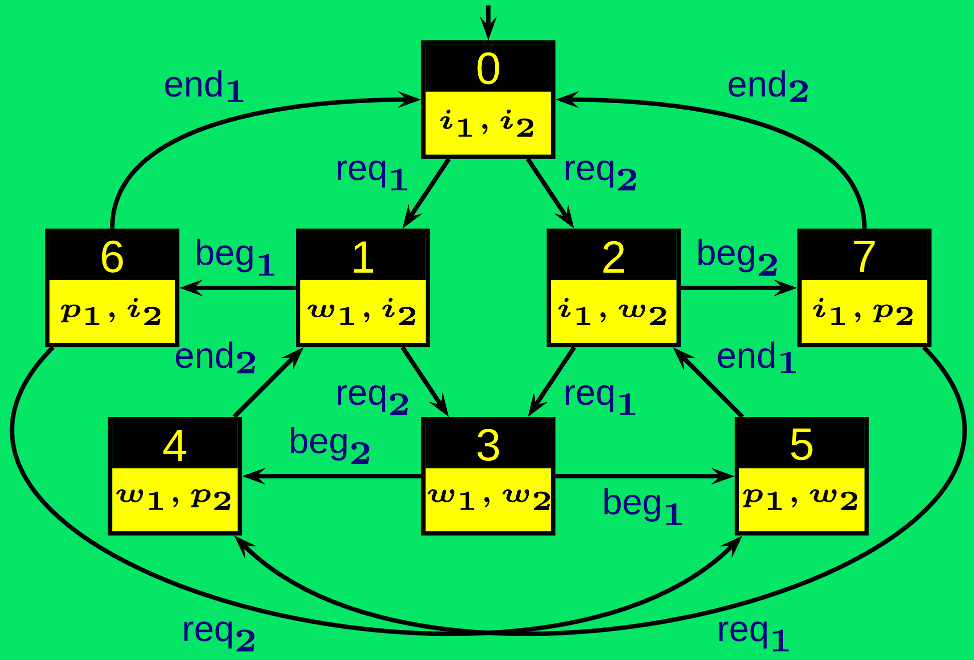
- 1 Printer, 1 Process
- Process can
  - stay idle
  - wait to print
  - print

→ Propositions i, w, p

- Actions of the printer:
  - req: User submits print request
  - beg: Start printing
  - end: End printing
- A property: $\mathbf{G}(w \longrightarrow \mathbf{F}p)$



A printer and two processes



$$\mathbf{G}[(w_1 \longrightarrow \mathbf{F}p_1) \wedge (w_2 \longrightarrow \mathbf{F}p_2)]$$

- k processes: 3^k states, growing formula size
- arbitrary number of processes?

The Model Checking Approach to Verification

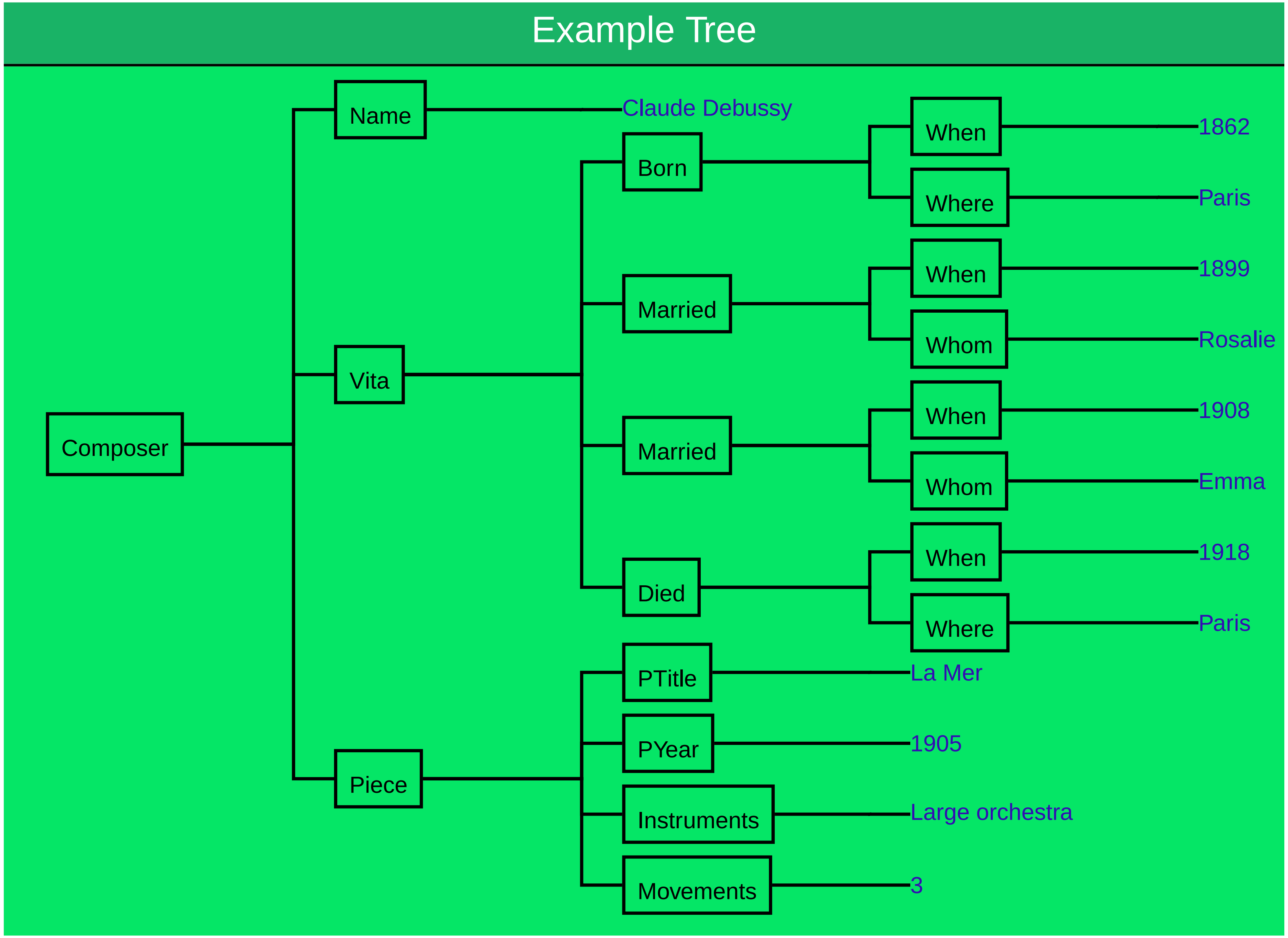
- Model checking:
  - System: M
  - Property: φ
  - Does M fulfil property φ?
- Model a ”real life” system as a transition system with finite state space

  ➞ Abstract away data values, process numbers,...
- Model executions of the system as infinite strings or trees
- Specify properties in a logic that allows translation into automata
- Use decidability of non-emptiness for automata to obtain decidability of model checking

---

- But sometimes this approach does not work, e.g., if number of processes is not bounded a priori

XML


Example Tree
Composer
Name
Claude Debussy
Vita
Born
When
1862
Where
Paris
Married
When
1899
Whom
Rosalie
Married
When
1908
Whom
Emma
Died
When
1918
Where
Paris
Piece
PTitle
La Mer
PYear
1905
Instruments
Large orchestra
Movements
3

XML Processing

Four important kinds of XML processing .... and their languages

Validation `DTD`, `XML Schema`

Check whether an XML document is of a given type

Navigation `XPath`

Select a set of positions in an XML document

Querying `XQuery`

Extract information from an XML document

Transformation `XSLT`

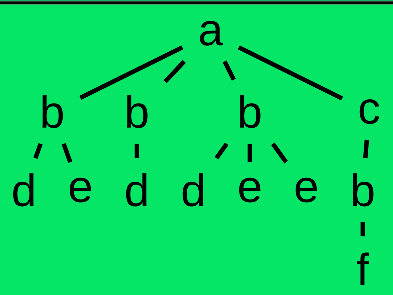
Construct a new XML document from a given one

XML as Tree

Example Document

⟨a⟩⟨b⟩
⟨d⟩12⟨/d⟩⟨e⟩22⟨/e⟩
⟨/b⟩
⟨b⟩⟨d⟩4⟨/d⟩⟨/b⟩
⟨b⟩⟨d⟩11⟨/d⟩
⟨e⟩8⟨/e⟩18⟨e⟩5⟨/e⟩
⟨/b⟩
⟨c⟩⟨b⟩⟨f⟩2⟨/f⟩
⟨/b⟩7⟨/c⟩
⟨/a⟩

...as Unranked Tree



- For many investigations, data values can and have to be ignored

→ Validation, Navigation

- Usually, XML trees are modeled as labeled trees over a finite alphabet
- For schema languages this is ok
- For queries, the actual alphabet might depend on the query
- Unary predicates on data values can be also modeled this way

The Automata Approach to XML Processing

- Many tasks concentrate on structure of XML documents and disregard data values

→ Model XML documents as unranked trees with a finite set of labels

→ Validation can be modelled by tree automata

→ Transformation can be modelled by tree transducers

- Make use of connection between FO or MSO logic and automata

→ Decidable static analysis

- But sometimes this approach does not work, e.g., key constraints can not be modelled

Summing up

- In verification and XML processing, finite alphabet strings and trees are used as approximations for the ”real world”
- This is sufficient for many applications
- It allows to exploit the logic-automata connection to obtain decidable Model Checking and static analysis

---

- Sometimes this modelling becomes messy
- Sometimes it is simply not good enough

- Goal of this research: Find logics and automata that allow for decidability even if data values are present
- Here: consider classical logics

Contents

▷ **Preliminaries**

Background on two variable logics

Two variable logics on data strings

Two variable logics on data trees

Conclusion

Our setting

Example: data string

17	5	2	3	3	3	2	2	7	17	17	3	4	5	2	3	3	4	4
a	b	c	b	c	a	a	b	b	b	c	a	b	a	c	b	b	a	a

Definition

- Data string: Finite sequence over $\Sigma \times D$, where
  - Σ finite
  - D infinite
- Our logical language:
  - $a(x)$: Letter position x is $a \in \Sigma$
  - order relation $<$, successor relation $+1$
  - $\sim$: $x \sim y$ if positions x and y have the same D-value

  → Equivalence relation

Definition

- Data tree: Tree with labels from Σ and one data value from $\mathcal{D}$ per node
- Logical language:
  - $\sim$: as for strings
  - $E_{\rightarrow}$: horizontal neighbor ("next sibling")
  - $E_{\downarrow}$: parent-child
  - $E_{\Rightarrow}$, $E_{\Downarrow}$: the transitive closures of $E_{\rightarrow}$ and $E_{\downarrow}$

Note: no other operations on data values, in particular no arithmetic

A first attempt

- We know:
  - First-order logic is undecidable in general
  - First-order logic is decidable on strings
- What about First-order logic on data strings?

Theorem

Satisfiability of First-Order formulas on data strings is undecidable, even for formulas with 3 variables

Proof idea

- Reduction from PCP:
  - Given: $(u_1, v_1), \ldots, (u_k, v_k)$, pairs of strings
  - Question: is there a sequence $i_1, \ldots, i_n$ such that $u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots u_{i_n}$?

A bit more detail

- Encode solution candidates as data strings over $\{a, b, \#, 1, \ldots, n\}$ of the form $u\#v$
- Each occurrence of a u_i is prefixed by i, i.e.:
  If $u_1 = aba$ and $u_2 = bb$ then 121 induces $1aba2bb1aba$
- Each data value occurs exactly twice, once in u and once in v

➞ corresponding positions should have the same data value (and same number/symbol)

- Crucial: check that sequence of data values is the same on both sides for number positions and letter positions

➞ Important subformula:

$$x \sim y \quad \rightarrow \quad \exists z\, (x + 1 = z \wedge \exists x\, y + 1 = x \wedge z \sim x)$$

"if x and y are equivalent then their right neighbors are also equivalent"

Two variables: a useful restriction?

- Classical approach: restrict to 2 variables
- Does this give us anything useful?

(1) We do not have free choice...

(2) We can express a lot of useful things with only two variables

...in the verification context

- whenever a print is requested by a process, it is granted at some point:

$$\forall x \exists y \,\mathsf{req}(x) \rightarrow (x < y \wedge \mathsf{beg}(y) \wedge x \sim y)$$

- No printer job is requested twice:

$$\forall x \forall y \,(\mathsf{req}(x) \wedge \mathsf{req}(y) \wedge x \sim y) \rightarrow x = y$$

...in the XML context

- With 2 variables many integrity constraints can be expressed:
  - Did we reserve a room for every participant?
  - $\forall x \; \mathsf{Partic.Name}(x) \rightarrow \exists y \mathsf{Room.Name}(y) \wedge x.\mathsf{data} = y.\mathsf{data}$
  - Does every participant give at most one talk?
  - $\forall x \forall y \; [\mathsf{Talk.Speaker}(x) \wedge \mathsf{Talk.Speaker}(y)] \rightarrow x.\mathsf{data} = y.\mathsf{data}$
- Close relationship with navigation: `XPath`
- Regular languages require only two variables (plus ex. quantification of sets)

Contents

Preliminaries

▷ **Background on two variable logics**

Two variable logics on data strings

Two variable logics on data trees

Conclusion

Some Known results about Two-Variable Logics

Theorem

- Two-variable logics has the finite model property [Mortimer 75]
- It actually has the exponential-size model property [Grädel, Otto 97]

→ Satisfiability in **NEXPTIME**

Proof idea: Step 1

- Scott Normal Form:

  With respect to satisfiability each **FO**^{2}-formula φ is equivalent to a formula φ' of the form:

  $$\forall x \forall y\, \chi \;\wedge\; \bigwedge_i \forall x \exists y\, \chi_i$$

  χ, χ_i quantifier-free

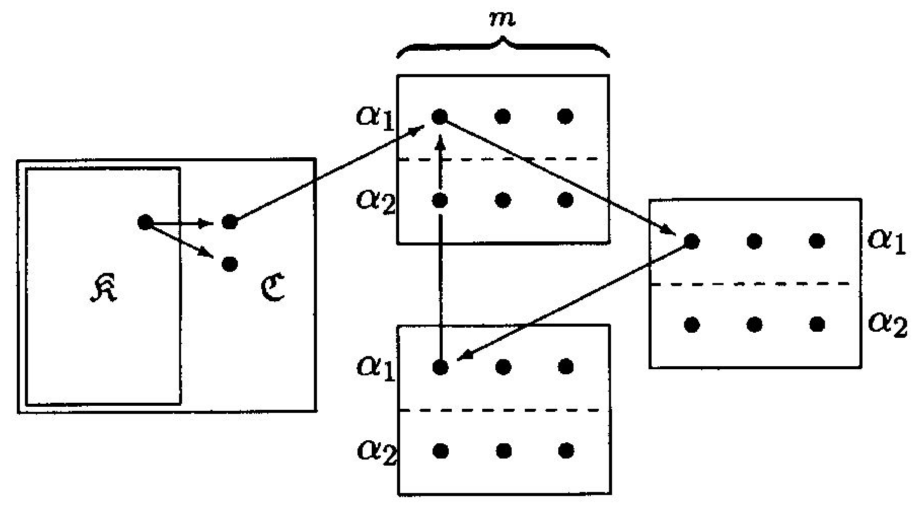
- φ' might have enlarged signature: more unary relations
- φ' can be computed in polynomial time

Proof idea: Step 2

- Assume there is a model $\mathcal{A}$ of φ'

→ Construct a model $\mathcal{B}$ for φ' of exp. size

- Consider basic 1-types and 2-types
- King: 1-type that occurs exactly once
- Construct $\mathcal{B}$ as follows:
  - Take the kings
  - Take χ_i-witnesses for the kings (Court)
  - Take three sets of witnesses and combine them in a circular fashion



- (Proof and picture: [Grädel, Otto 97])

Is this the end of the story?

- Satisfiabilty of **FO^2** decidable
- Is this applicable to our setting?
- Unfortunately not: on structures with particular properties the situation is more complicated

More Results on general structures

- On structures with 1 or 2 equivalence relations: decidable [Kieroński, Otto 05]
- On structures with 3 equivalence relations: undecidable [Kieroński, Otto 05]
- On structures with a linear order: in **coNEXPTIME** [Otto]
- On structures with several well-orderings: undecidable [Otto]

...on strings without data

- Satisfiability **NEXPTIME**-complete
- Expressive power:
  - unary LTL and $\Sigma^2 \cap \Pi^2$ [Etessami, Vardi, Wilke 97]
  - variety DA [Thérien, Wilke 98]

More known results

- Regular string languages: EMSO with two FO-variables (in the presence of $+1$)
- Regular tree languages: EMSO with two FO-variables (in the presence of $E_{\rightarrow}$, $E_{\downarrow}$)

Contents

Preliminaries

Background on two variable logics

▷ **Two variable logics on data strings**

Two variable logics on data trees

Conclusion

Some notation

Some Definitions

- Data string s:

Class

Interval

$$\begin{array}{ccccccccccccccccccc}
\mathbf{17} & \mathbf{5} & \mathbf{2} & \mathbf{3} & \mathbf{3} & \mathbf{3} & \mathbf{2} & \mathbf{2} & \mathbf{7} & \mathbf{17} & \mathbf{17} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{2} & \mathbf{3} & \mathbf{3} & \mathbf{4} & \mathbf{4} \\
a & b & c & b & c & a & a & b & b & b & c & a & b & a & c & b & b & a & a
\end{array}$$

- **Class**: all positions with the same data value
- **Interval**: (maximal set of) contiguous positions of a class
- **String projection $P(s)$**: $abcbcaabbbcabacbbaa$

- **type $\alpha(x)$**: any conjunction of unary literals

On the expressive power of FO^2 on data strings

Example

- φ_a:
  - $\varphi_a = \forall x \forall y (x \neq y \wedge a(x) \wedge a(y)) \rightarrow x \not\sim y$
  - all a's are in different classes
- Similarly: φ_b
- $\psi_{a,b}$:
  - $\psi_{a,b} = \forall x \exists y \big(a(x) \rightarrow (b(y) \wedge x \sim y)\big)$
  - each class with an a also contains a b
- Similarly: $\psi_{b,a}$.

- Thus:
  - $\varphi = \varphi_a \wedge \varphi_b \wedge \psi_{a,b} \wedge \psi_{b,a}$
  - the numbers of a and b-labeled positions are equal
- In a similar fashion: number of a's, b's and c's are equal

➞ non-regular language

On the expressive power of FO^2 on data strings (cont.)

More example properties

- Let α and β denote unary types
- FO^2 can express
  - data-blind properties, i.e., properties not using $\sim$
  - All occurrences of a type α are in the same class:

$$\theta = \forall x \forall y \left((\alpha(x) \land \alpha(y)) \rightarrow x \sim y \right)$$

  - Each class contains at most one occurrence of α:

$$\theta = \forall x \forall y \left((\alpha(x) \land \alpha(y) \land x \sim y) \rightarrow x = y \right)$$

  - In each class, every α occurs before every β:

$$\theta = \forall x \forall y \left((\alpha(x) \land \beta(y) \land x \sim y) \rightarrow x < y \right)$$

  - Each class with an α also has a β:

$$\theta = \forall x \exists y \left(\alpha(x) \rightarrow (\beta(y) \land x \sim y) \right)$$

  - That's all!

Main result regarding data strings

Theorem

Satisfiability of **FO**$^2(+1, <)$ on data strings is decidable

Main steps of the proof

FO^2 formula φ

$\downarrow$

Scott normal form

$\downarrow$

Intermediate normal form

$\downarrow$

Data normal form ψ

$\downarrow$

Data automaton $\mathcal{D}_\psi$

$\downarrow$

Multicounter automaton $\mathcal{A}_\psi$ (on strings)

- Finally: $\mathcal{A}_\psi$ accepts a string w if and only if there is a data string s with
  - $s \models \varphi$
  - $P(s) = w$
- Thus: ψ satisfiable $\Longleftrightarrow L(\mathcal{A}_\psi) \neq \emptyset$

Steps 1 and 2

Scott and intermediate normal form

- We transform into satisfiability equivalent EMSO formulas
- **Scott normal form**: $\exists R_1, \ldots, R_k \; \forall x \forall y \, \chi \;\wedge\; \bigwedge_i \forall x \exists y \, \chi_i$
- **Intermediate normal form**:

$$\exists R_1 \cdots R_m \theta_1 \wedge \cdots \wedge \theta_n$$

- θ_i:

(1) $\forall x \forall y \quad (\delta(x,y) \geq 2 \wedge \alpha(x) \wedge \beta(y) \wedge \boxed{\begin{array}{c} x \sim y \\ x \not\sim y \end{array}}) \rightarrow \boxed{\begin{array}{c} x < y \\ x > y \\ \mathit{ff} \end{array}}$

(2) $\forall x \exists y \quad \alpha(x) \rightarrow (\beta(y) \wedge \boxed{\begin{array}{c} x \sim y \\ x \not\sim y \end{array}} \wedge \boxed{\begin{array}{c} x+1 < y \\ x+1 = y \\ x = y \\ x = y+1 \\ x > y+1 \end{array}})$

(3) $\forall x \forall y \quad \theta$ (quantifier-free, DNF, no $\sim$)

- Both steps are straightforward

Step 3

Data normal form

- **Data normal form**: Disjunction of formulas

$$\exists R_1 \cdots R_n \; \theta_1 \wedge \cdots \wedge \theta_n$$

- θ_i:

  (a) data-blind

  (b) All α are in the same class

  (c) Each class contains at most one α

  (d) In each class, every α occurs before every β

  (e) Each class with an α also has a β

  (f) If x is in a different class than its successor it has type α

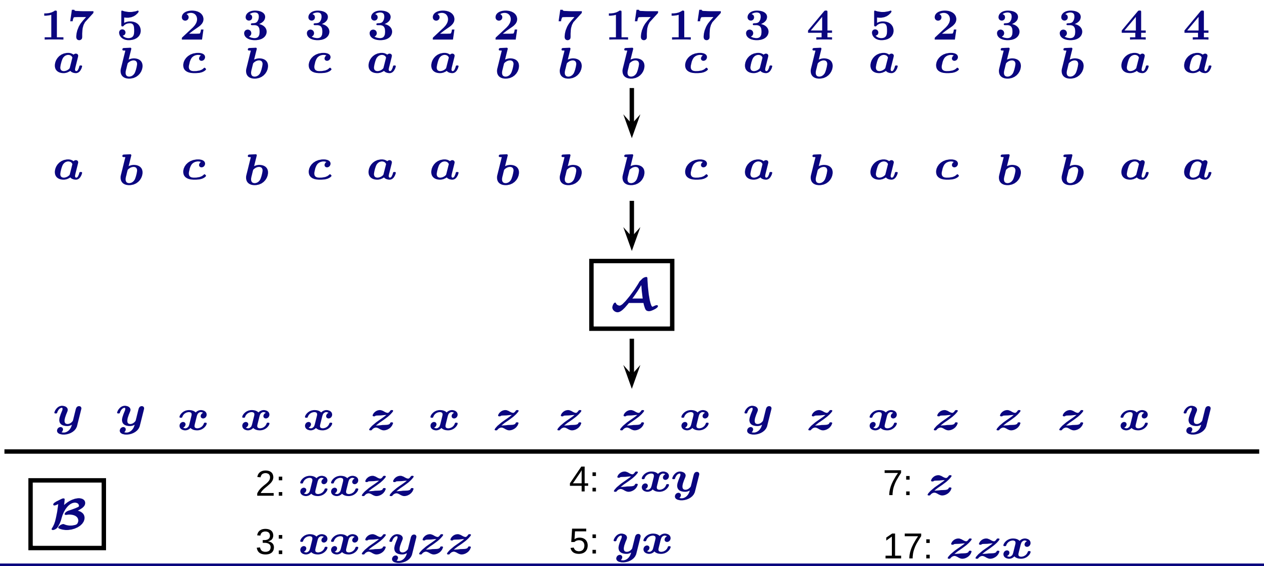
- For each type α we capture the two left-most classes with α and the two rightmost classes with α by unary relations $R_1^\alpha, \ldots, R_4^\alpha$
- Case distinction on possible formulas (1) and (2)

→ in each case θ_i can be replaced by some "data normal" formulas

Step 4

Data automata

- A **data automaton** $\mathcal{D} = (\mathcal{A}, \mathcal{B})$ consists of a
  - non-deterministic letter-to-letter string transducer $\mathcal{A}$ (**base automaton**) and a
  - string automaton $\mathcal{B}$ (**class automaton**)
- A data word w is **accepted** by $\mathcal{D}$ if
  - $\mathcal{A}$ on input $P(w)$ accepts with some output $b_1 \cdots b_n$ such that
  - for *each class* $\{x_1, \ldots, x_k, \ldots, \subseteq\}\{1, \ldots, n, \ldots, \}$, $x_1 < \cdots < x_k$, $\mathcal{B}$ accepts $b_{x_1}, \ldots, b_{x_k}$.



Proposition

For each two-variable formula over data strings there is an equivalent data automaton.

Step 5

Multicounter-automata

- A **multicounter-automaton** is a finite automaton with counters $C_1, \ldots, C_k$
- Counters can be incremented and decremented (if a counter is zero and decremented, computation stops)
- A run is accepting, if at the end, all counters are zero

Theorem [Mayr 84, Kosaraju 84]

Emptiness for multicounter automata is decidable (complexity-wise equivalent to Petri net reachability)

Proposition

For each data automaton $\mathcal{D} = (\mathcal{A}, \mathcal{B})$ there is a multicounter automaton $\mathcal{M}$ such that $L(\mathcal{M}) = P(L(\mathcal{D}))$

Proof idea

- Plan: $\mathcal{M}$ checks whether a given string w can be extended to a data string w' accepted by $\mathcal{D}$
- Clearly: $\mathcal{M}$ can simulate transducer $\mathcal{A}$ (which does not see the data anyway)
- Idea: $\mathcal{M}$ uses one counter C_p, for each state p of $\mathcal{B}$
- After reading a prefix u of v, counter C_p tells in how many class strings state p is reached
- At the end: all C_p for non-accepting p must be zero

Related results

- Complexity of reductions:

Satisfiability for **FO**$(<, +1)$ on data words

2NEXPTIME $\downarrow\uparrow$ PTIME

Emptiness for data automata

PTIME $\downarrow\uparrow$ PTIME

Reachability for Petri nets

→ Complexity of Satisfiability for **FO**$(<, +1)$ on data words closely related to the unknown complexity of Reachability for Petri nets (no elementary upper bound known)

- Class successor: $i \pm 1 = j$ iff
  - $i < j$, $i \sim j$, and
  - no $i < k < j$, $i \sim k$

Proposition

A language is recognizable by a data automaton iff it is definable in $\mathbf{EMSO}^2(\sim,<,+1,\pm 1)$

Proposition

- Satisfiability of $\mathbf{FO}^2(\sim,<)$ on data words is **NEXPTIME**-complete
- Satisfiability of $\mathbf{FO}^2(\sim,+1)$ on data words is $2-$ **NEXPTIME**-complete

- Decidability remains if predicates $+2$, $+3$,... are added
- Results generalize to ω-strings

- Undecidable, e.g. with:
  - 3 variables,
  - 2 equivalence classes, or
  - linear order on data values

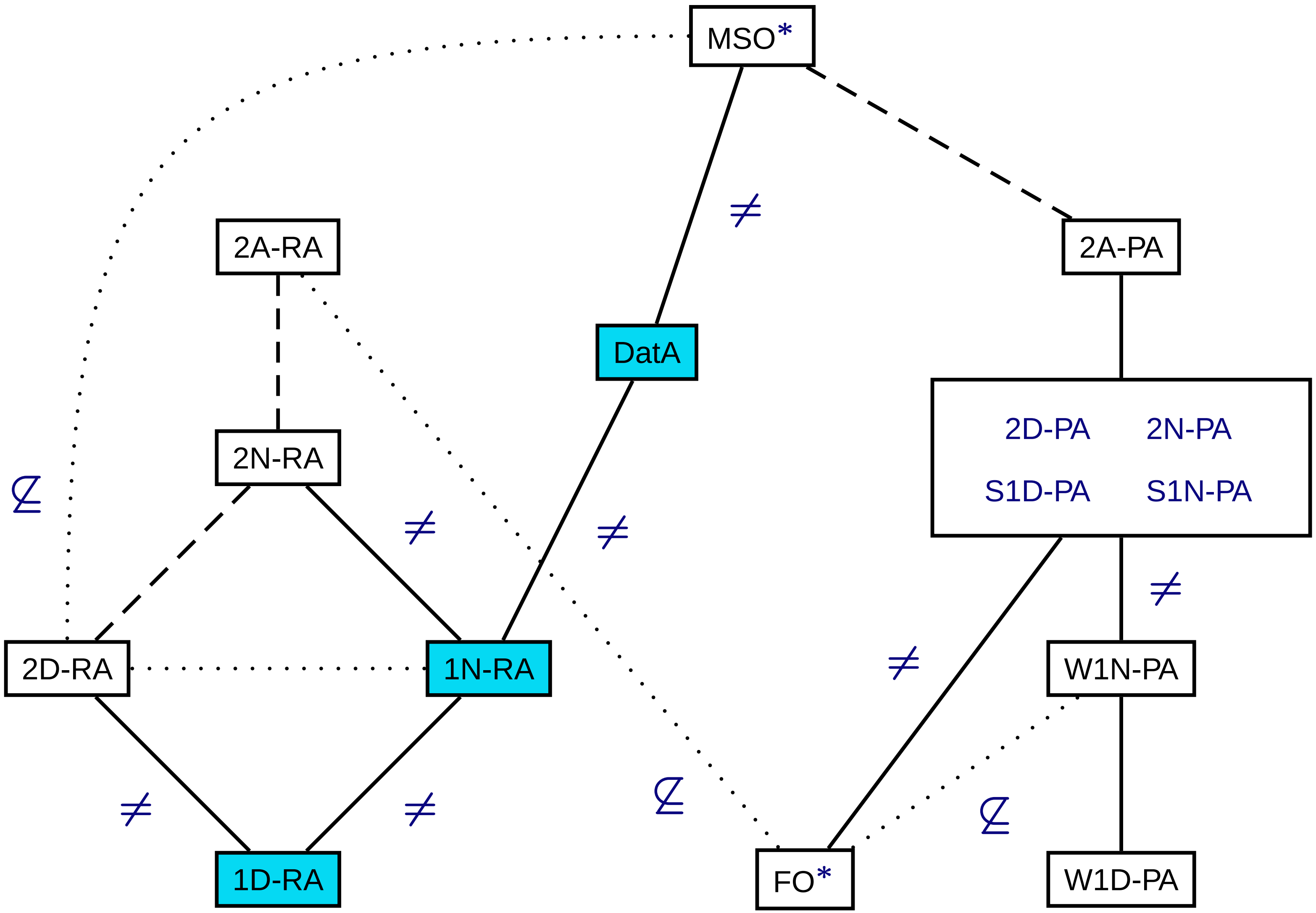
- Temporal logics for data strings:
  - Freeze operator [Demri, Lazić 06]
  - ongoing work: other data extensions of temporal logics

A related issue

- Regular string and tree languages play a key role for decidability results
- Natural question: Is there a sensible notion of regular languages for data strings?

- Register automata [Kaminski, Francez 94]
- Regular expressions [Kaminski, Tan 04]
- Pebble automata [Neven, Sch., Vianu 01]
- General picture: most models pairwise different and/or undecidable

Inclusion structure of Automata Models



Regular data languages?

- Maybe there is not THE class of regular data languages
- But: the class of languages accepted by data automata is pretty strong and robust

---

- A different view: just a finite automaton reading $P(w)$, but transitions depend on
  - state at left neighbor, and
  - state at class predecessor

➞ allows for deterministic automata model (weaker than non-deterministic)

- (joint work with Henrik Björklund)

Contents

Preliminaries

Background on two variable logics

Two variable logics on data strings

▷ **Two variable logics on data trees**

Conclusion

Coming soon...

13. Jahrestagung der GI-Fachgruppe

Logik in der Informatik

Dortmund, 12.-13. Oktober 2006

Invited speaker:

Anuj Dawar

More infos:

loginf.cs.uni-dortmund.de

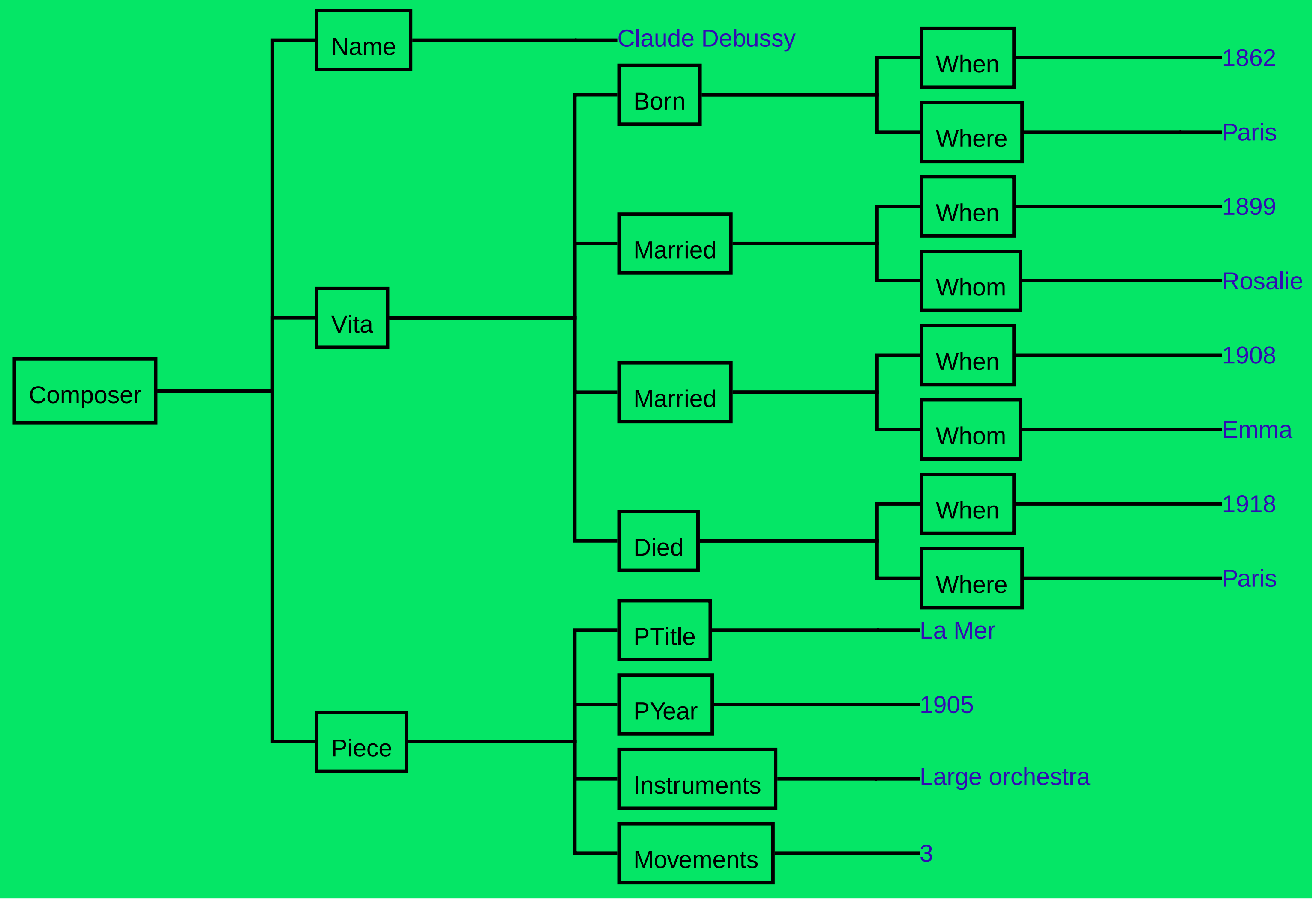
Data trees: notation

Definition

- Data tree: Tree with labels from Σ and one data value from $\mathcal{D}$ per node
- Logical language:
  - $\sim$: as for strings
  - $E_{\rightarrow}$: horizontal neighbor (”next sibling”)
  - $E_{\downarrow}$: parent-child
  - $E_{\Rightarrow}$, $E_{\Downarrow}$: the transitive closures of $E_{\rightarrow}$ and $E_{\downarrow}$
- $+1$: $E_{\rightarrow}$ and $E_{\downarrow}$
- $<$: $E_{\Rightarrow}$ and $E_{\Downarrow}$
- $\mathbf{FO}^2(\sim,+1)$: only next sibling and child
- $\mathbf{FO}^2(\sim,+1,<)$: also: descendant and following sibling

Back to XML applications

Example



XML integrity constraints

- **key constraint:** $\tau[X] \to \tau$

  X-attributes determine a node
- **inclusion constraint**: $\tau[X] \subseteq \tau'[Y]$

  X-values of τ-nodes occur as Y-values of τ'-nodes
- Important reasoning tasks:
  - Consistency of integrity constraints
  - Implication of integrity constraints
- Unary key constraints and inclusion constraints can be expressed with 2 variables

→ From our decidability results we can conclude a wide range of decidability results for integrity constraints

- Related work: [Arenas, Fan, Libkin 05; Buneman et al., 03]

- Furthermore: regular tree languages can be captured by $\mathbf{EMSO^2(\sim,+1)}$
- XML schema languages correspond to regular tree languages

→ Reasoning on integrity constraints relative to schema descriptions possible

→ in particular: stronger notions of types can be handled

Theorem

The consistency and implication problems for unary keys and unary inclusion constraints relative to a regular tree language are decidable

$\texttt{XPath}$ containment

- $\texttt{XPath}$ is a navigation language for $\texttt{XML}$
- In its basic form it selects a set of nodes in a tree, which can be reached from the root on a path with certain properties
- Navigation along different axes
- The core of $\texttt{XPath}$ (ignoring data values) corresponds exactly to $\mathbf{FO}^2(\sim,+1,<)$ [Marx 05]
- Query-Containment for $\texttt{XPath}$ is in **EXPTIME** [Marx 05]

- Our setting allows to consider a limited form of tests on attribute values, e.g.:

  Child:a@A = /Child:b@C

- A larger example:

$$\texttt{Child} :: a/\texttt{Child} :: b/@B_1 =$$
$$/\texttt{Child} :: c/\texttt{NextSibling} :: d/$$

translates to

$$\exists y E_\downarrow(x,y) \wedge a(y) \wedge$$
$$\exists x E_\downarrow(y,x) \wedge b(x) \wedge$$
$$\exists y E_\downarrow(x,y) \wedge B_1(y) \wedge$$
$$\exists x\, x \sim y \wedge B_2(x) \wedge$$
$$\exists y E_\downarrow(y,x) \wedge d(y) \wedge$$
$$\exists x E_\rightarrow(x,y) \wedge c(x) \wedge$$
$$\exists y E_\downarrow(y,x) \wedge \neg\exists x E_\downarrow(x,y)$$

Theorem

Query Containment for special $\texttt{XPath}$ is decidable

The general case: $\mathrm{FO}^2(\sim,+1,<)$ on data trees

- **Vector addition tree automata**: automata for binary trees
- Run induces a state and a vector of natural numbers, for each node
- Transitions $(q, l, q_0, q_1, \vec{a}, \vec{b}, \vec{c})$: automaton can take configuration $(q, \vec{z})$ at v if
  - v has label l
  - children have configurations $(q_0, \vec{x})$ and $(q_1, \vec{y})$
  - $\vec{x} - \vec{a} \geq \vec{0}$ and $\vec{y} - \vec{b} \geq \vec{0}$,
  - $\vec{z} = \vec{x} - \vec{a} + \vec{y} - \vec{b} + \vec{c}$

- Decidability of emptiness of vector addition tree automata is an open problem
- Equivalent to decidability of Multiplicative Exponential Linear Logic

Theorem

For any vector addition tree automaton A, a formula $\varphi_A \in \mathrm{FO}^2(\sim,<,+1)$ can be computed such that $L(A) \neq \emptyset$ iff φ_A has a model.

➞ We concentrate on $\mathrm{FO}^2(\sim,+1)$

Main result and proof structure

Theorem

Satisfiability of **FO**$^2(+1,<)$ on data strings is decidable

Main steps of the string proof

FO^2 formula φ

$\downarrow$

Scott normal form

$\downarrow$

Intermediate normal form

$\downarrow$

Data normal form ψ

$\downarrow$

Data automaton $\mathcal{D}_\psi$

$\downarrow$

Multicounter automaton $\mathcal{A}_\psi$ (on strings)

Theorem

Satisfiability of **FO**$^2(+1)$ on data trees is decidable

Main steps of the tree proof

FO^2 formula φ

$\downarrow$

Scott normal form

$\downarrow$

Intermediate normal form

$\downarrow$

Data normal form ψ

$\downarrow$

Puzzle P_ψ

$\downarrow$

Small model property

$\downarrow$

Linear constraint tree automaton $\mathcal{A}_\psi$ (on trees)

Steps 1 and 2

- Scott normal form
- Intermediate normal form:

$$\exists R_1 \cdots R_m \theta_1 \wedge \cdots \wedge \theta_n$$

- θ_i:

(1) $\forall x \forall y \quad \left[\alpha(x) \ \wedge \ \beta(y) \ \wedge \gamma(x,y)\right] \ \rightarrow (\delta(x,y) \leq 1)$

(2) $\forall x \exists y \quad \alpha(x) \rightarrow \ \left[\beta(y) \ \wedge \ \gamma(x,y) \ \wedge \ \epsilon(x,y)\right]$

- where
  - α, β are types.
  - $\gamma(x,y)$ is either $x \sim y$ or $x \not\sim y$, and
  - $\epsilon(x,y)$ is one of $E_\downarrow(x,y)$, $E_\downarrow(y,x)$, $E_\rightarrow(x,y)$, $E_\rightarrow(y,x)$, $x = y$, or $(\delta(x,y) > 1)$.

Step 3 and 4

- **Data normal form**:
$$\exists R_1 \cdots R_n\ \theta_1 \wedge \cdots \wedge \theta_n$$
- θ_i:
  (a) data-blind
  (b) “Each class contains at most one node of type α.”
  (c) “Each class with at least one node of type α has no β.”
  (d) “Each class with at least one node of type α also has a β.”
  (e) “Each node of type α has profile p”.
- (Profile: data equality type wrt. parent, left and right sibling)

- **Puzzle**: regular data-blind condition R and pairs $(D_1, S_n), \ldots, (D_k, S_k)$ of types
- A data tree is a **solution** to a puzzle, if
  - its data-free projection fulfils P, and
  - for each class C there is some i such that
    * each type from D_i occurs exactly once in C, and
    * all other types in C are from S_i
  - From a $\mathbf{FO}^2(\sim, +1)$ formula ϕ one can construct a satisfiability equivalent puzzle P_φ

Step 5

- Zone : maximally connected sub-tree within one (data) class
- degree of a zone: number of neighboring classes

Proposition

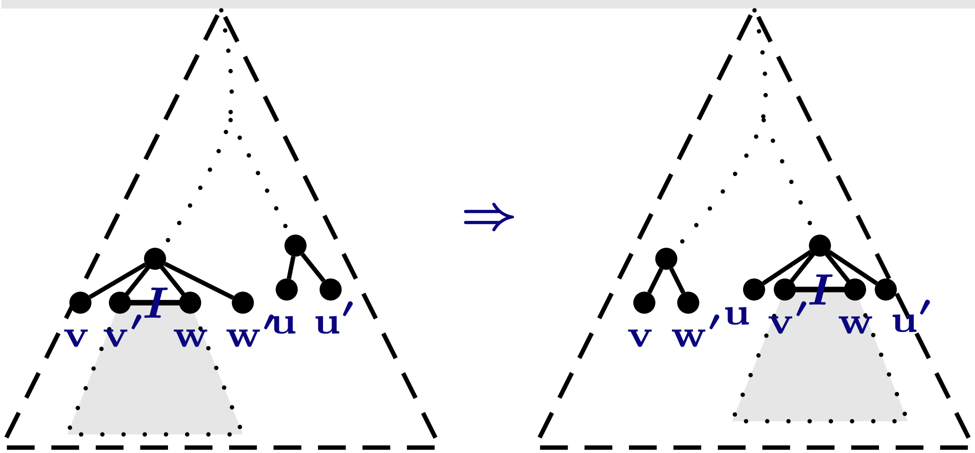
For every puzzle $\mathcal{P}$, one can compute constants M, N such that $\mathcal{P}$ has a solution only if it has a solution in which at most M zones have degree more than N.

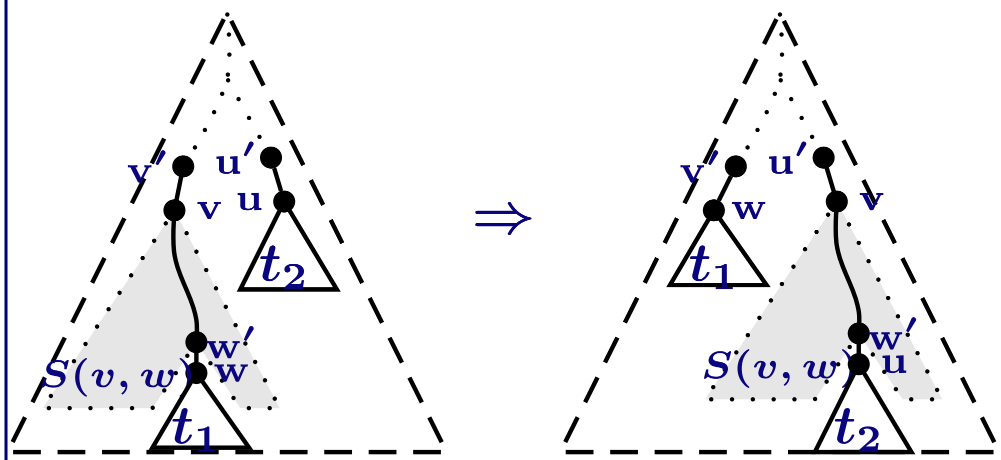
Proof strategy

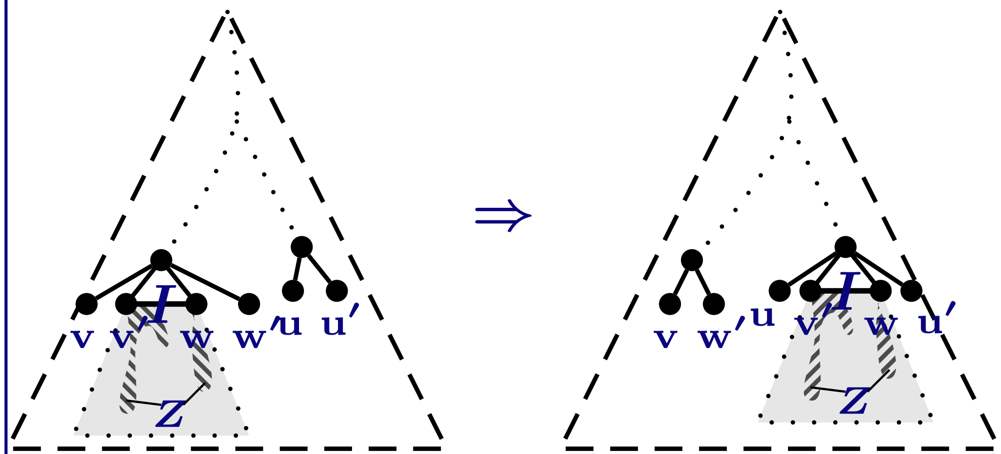
- We first show a local version of the proposition which talks about each class:

→ local pruning
- For the general proposition:

→ global pruning
- We use (tons of) cut-and-paste arguments







Step 6

- **Linear constraint tree automaton (LCTA)**: unranked tree automaton $\mathcal{A}$ with state set Q and linear constraints over variables $\#_q$

  ($\#_q$: number of nodes with state q)

Proposition
Non-emptiness of LCTA is **NP**-complete

Proposition
Let P be a puzzle over Σ and let $M, N \in \mathbb{N}$. There is an LCTA that recognizes the data erasure of solutions of P where at most M zones have degree more than N.

- This completes the proof that $\mathbf{FO^2}(\sim, +1)$ on data trees is decidable
- Complexity:
  - upper bound: 3-**NEXPTIME**
  - lower bound: **NEXPTIME**

Contents

Preliminaries

Background on two variable logics

Two variable logics on data strings

Two variable logics on data trees

▷ **Conclusion**

Conclusion

- Two-variable logic on data strings and data trees offers an interesting framework to obtain decidability results in verification and XML reasoning
- Two main results:
  - $\mathbf{FO^2(\sim, <, +1)}$ decidable on data strings
  - $\mathbf{FO^2(\sim, +1)}$ decidable on data trees

- Open problems:
  - $\mathbf{FO^2(\sim, <, +1)}$ on data trees
  - $\mathbf{FO^2(\sim, +\omega)}$ on data trees
  - Settle the open complexity issues
  - Find tractable fragments
  - What are regular data string languages / regular data tree languages
  - Explore applicability of results in verification

- References:

  - Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, Claire David: Two-Variable Logic on Words with Data. LICS 2006: 7-16
  - Mikolaj Bojanczyk, Claire David, Anca Muscholl, Thomas Schwentick, Luc Segoufin: Two-variable logic on data trees and XML reasoning. PODS 2006: 10-19

Finally...

Thank you!