

TQL and Tree Automata

Automata and Logics for Unordered Trees

Jean-Marc Talbot
Joint work with Iovka Boneva and Sophie Tison

LIFL - STC Team
INRIA Futurs - MOSTRARE Project

Jun. 9th 2005



Outline

1 Preliminaries

- The Tree Model
- Monadic second-order logic and some extensions

2 The Tree Logic TQL

- An Algebraic View of Trees
- The Fragment TQL^μ

3 Decision Problems for TQL^μ

- The Model-Checking Problem
- The Satisfiability Problem
- Expressiveness of TQL^μ

Outline

1 Preliminaries

- The Tree Model
- Monadic second-order logic and some extensions

2 The Tree Logic TQL

- An Algebraic View of Trees
- The Fragment TQL^μ

3 Decision Problems for TQL^μ

- The Model-Checking Problem
- The Satisfiability Problem
- Expressiveness of TQL^μ

Outline

1 Preliminaries

- The Tree Model
- Monadic second-order logic and some extensions

2 The Tree Logic TQL

- An Algebraic View of Trees
- The Fragment TQL ^{μ}

3 Decision Problems for TQL ^{μ}

- The Model-Checking Problem
- The Satisfiability Problem
- Expressiveness of TQL ^{μ}

Trees

Let Λ be a finite set of labels

Definition

A **tree** τ is a graph (V, E, λ) such that

- V is a finite non-empty set of nodes
- $E \subseteq V \times V$ is a set of edges
- λ is a mapping from E to Λ (edge-labeled tree)
- V contains a distinguished node r (the root of the tree) and there exists a unique path from r to v , for any node v in V

Trees

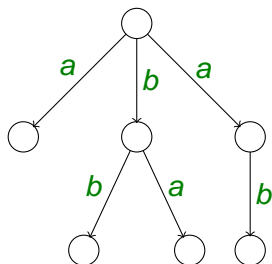
Let Λ be a finite set of labels

Definition

A **tree** τ is a graph (V, E, λ) such that

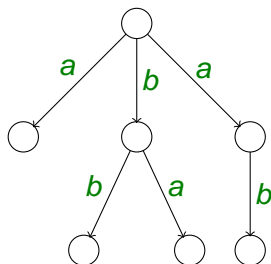
- V is a finite non-empty set of nodes
- $E \subseteq V \times V$ is a set of edges
- λ is a mapping from E to Λ (edge-labeled tree)
- V contains a distinguished node r (the root of the tree) and there exists a unique path from r to v , for any node v in V

Unranked and Unordered Trees



- no bound on the number of children for nodes (**unranked tree**)
- no relation between children of a node (**unordered tree**)

Unranked and Unordered Trees



- no bound on the number of children for nodes (**unranked tree**)
- no relation between children of a node (**unordered tree**)

Trees as Relational Structures

We consider the signature $\Sigma = \{\text{label}_a \mid a \in \Lambda\}$.

With any tree $\tau = (V, E, \lambda)$, one associates the structure \mathcal{S}^τ over Σ defined as:

- the domain of \mathcal{S}^τ is V
- $\text{label}_a^\tau(v_1, v_2)$ if $(v_1, v_2) \in E$ and $\lambda((v_1, v_2)) = a$

MSO : Syntax and Semantics

Given x, y, z, \dots first-order variables
 X, Y, Z, \dots (monadic) second-order variables

Syntax: a formula φ

$$\varphi ::= \text{label}_a(x, y) \mid x \in X \mid \varphi \vee \varphi \mid \neg \varphi \mid \exists x. \varphi \mid \exists X. \varphi$$

Given a valuation σ with $\sigma(x) \in V$ and $\sigma(X) \in \wp(V)$

Semantics: $\mathcal{S}, \sigma \models \varphi$ if

$\text{label}_a(x, y)$	if $\text{label}_a(\sigma(x), \sigma(y))$ holds in \mathcal{S}
$x \in X$	if $\sigma(x)$ belongs to $\sigma(X)$
$\exists x. \varphi$	if there exists $s \in \text{dom}(\mathcal{S})$ st $\mathcal{S}, \sigma[x \rightarrow s] \models \varphi$
$\exists X. \varphi$	if there exists $S \subseteq \text{dom}(\mathcal{S})$ st $\mathcal{S}, \sigma[X \rightarrow S] \models \varphi$

MSO : Syntax and Semantics

Given x, y, z, \dots first-order variables
 X, Y, Z, \dots (monadic) second-order variables

Syntax: a formula φ

$$\varphi ::= \text{label}_a(x, y) \mid x \in X \mid \varphi \vee \varphi \mid \neg \varphi \mid \exists x. \varphi \mid \exists X. \varphi$$

Given a valuation σ with $\sigma(x) \in V$ and $\sigma(X) \in \wp(V)$

Semantics: $\mathcal{S}, \sigma \models \varphi$ if

$\text{label}_a(x, y)$	if $\text{label}_a(\sigma(x), \sigma(y))$ holds in \mathcal{S}
$x \in X$	if $\sigma(x)$ belongs to $\sigma(X)$
$\exists x. \varphi$	if there exists $s \in \text{dom}(\mathcal{S})$ st $\mathcal{S}, \sigma[x \rightarrow s] \models \varphi$
$\exists X. \varphi$	if there exists $S \subseteq \text{dom}(\mathcal{S})$ st $\mathcal{S}, \sigma[X \rightarrow S] \models \varphi$

MSO over Trees: some examples

- There is an edge labeled with a : $\exists x, y \text{ label}_a(x, y)$
- the root has at least two children :

$$\exists x \text{ root}(x) \wedge \exists y, z (\text{edge}(x, y) \wedge \text{edge}(x, z) \wedge z \neq y)$$

where $\text{edge}(x, y) = \bigvee_{a \in \Lambda} \text{label}_a(x, y)$ and $\text{root}(x) = \forall y (\neg \text{edge}(y, x))$

- there is a path from the root to a leaf labeled only by a :

$$\exists x \text{ root}(x) \wedge \forall P (x \in P \wedge (\forall z_1, z_2 (z_1 \in P \wedge \text{label}_a(z_1, z_2)) \rightarrow z_2 \in P) \rightarrow \exists y (y \in P \wedge \text{leaf}(y)))$$

where $\text{leaf}(x) = \forall y (\neg \text{edge}(x, y))$

BUT!!! over unranked and unordered trees, properties such as

- “the root has an even number of out-going edges”
- “the number of edges in the tree is even”

are not MSO-definable [Courcelle:IC90].

MSO over Trees: some examples

- There is an edge labeled with a : $\exists x, y \text{ label}_a(x, y)$
- the root has at least two children :

$$\exists x \text{ root}(x) \wedge \exists y, z (\text{edge}(x, y) \wedge \text{edge}(x, z) \wedge z \neq y))$$

where $\text{edge}(x, y) = \bigvee_{a \in \Lambda} \text{label}_a(x, y)$ and $\text{root}(x) = \forall y (\neg \text{edge}(y, x))$

- there is a path from the root to a leaf labeled only by a :

$$\exists x \text{ root}(x) \wedge \forall P (x \in P \wedge (\forall z_1, z_2 (z_1 \in P \wedge \text{label}_a(z_1, z_2)) \rightarrow z_2 \in P)) \rightarrow \exists y (y \in P \wedge \text{leaf}(y))$$

where $\text{leaf}(x) = \forall y (\neg \text{edge}(x, y))$

BUT!!! over unranked and unordered trees, properties such as

- “the root has an even number of out-going edges”
- “the number of edges in the tree is even”

are not MSO-definable [Courcelle:IC90].

MSO over Trees: some examples

- There is an edge labeled with a : $\exists x, y \text{ label}_a(x, y)$
- the root has at least two children :

$$\exists x \text{ root}(x) \wedge \exists y, z (\text{edge}(x, y) \wedge \text{edge}(x, z) \wedge z \neq y)$$

where $\text{edge}(x, y) = \bigvee_{a \in \Lambda} \text{label}_a(x, y)$ and $\text{root}(x) = \forall y (\neg \text{edge}(y, x))$

- there is a path from the root to a leaf labeled only by a :

$$\begin{aligned} \exists x \text{ root}(x) \wedge \forall P (x \in P \wedge (\forall z_1, z_2 (z_1 \in P \wedge \text{label}_a(z_1, z_2)) \rightarrow z_2 \in P)) \\ \rightarrow \exists y (y \in P \wedge \text{leaf}(y)) \end{aligned}$$

where $\text{leaf}(x) = \forall y (\neg \text{edge}(x, y))$

BUT!!! over unranked and unordered trees, properties such as

- “the root has an even number of out-going edges”
- “the number of edges in the tree is even”

are not MSO-definable [Courcelle:IC90].

MSO over Trees: some examples

- There is an edge labeled with a : $\exists x, y \text{ label}_a(x, y)$
- the root has at least two children :

$$\exists x \text{ root}(x) \wedge \exists y, z (\text{edge}(x, y) \wedge \text{edge}(x, z) \wedge z \neq y)$$

where $\text{edge}(x, y) = \bigvee_{a \in \Lambda} \text{label}_a(x, y)$ and $\text{root}(x) = \forall y (\neg \text{edge}(y, x))$

- there is a path from the root to a leaf labeled only by a :

$$\begin{aligned} \exists x \text{ root}(x) \wedge \forall P (x \in P \wedge (\forall z_1, z_2 (z_1 \in P \wedge \text{label}_a(z_1, z_2)) \rightarrow z_2 \in P)) \\ \rightarrow \exists y (y \in P \wedge \text{leaf}(y)) \end{aligned}$$

where $\text{leaf}(x) = \forall y (\neg \text{edge}(x, y))$

BUT!!! over unranked and unordered trees, properties such as

- “the root has an even number of out-going edges”
- “the number of edges in the tree is even”

are not MSO-definable [Courcelle:IC90].

MSO over Trees: Decision Problems

- **Satisfiability** for MSO over trees

Given a MSO sentence φ , decide whether there exists a tree τ such that

$$\mathcal{S}^\tau \models \varphi$$

The problem is *decidable* (but non-elementary) [Courcelle:IC90]

- **Model-checking** for MSO over trees

Given a tree τ and a MSO sentence φ , decide whether $\mathcal{S}^\tau \models \varphi$

- ▶ combined complexity : *PSPACE-complete*
- ▶ data complexity : *linear*

MSO over Trees: Decision Problems

- **Satisfiability** for MSO over trees

Given a MSO sentence φ , decide whether there exists a tree τ such that

$$\mathcal{S}^\tau \models \varphi$$

The problem is *decidable* (but non-elementary) [Courcelle:IC90]

- **Model-checking** for MSO over trees

Given a tree τ and a MSO sentence φ , decide whether $\mathcal{S}^\tau \models \varphi$

- ▶ combined complexity : *PSPACE-complete*
- ▶ data complexity : *linear*

Beyond MSO: Counting MSO [Courcelle:IC90]

$Mod_i^j(X)$ holds if $\text{card}(\sigma(X)) \bmod i = j$

- “the number of edges in the tree is even”

$$\exists X \forall x (x \in X \wedge Mod_0^2(X))$$

- “the root has an even number of out-going edges”

$$\forall X (\forall x (\exists y \text{ root}(y) \wedge \text{edge}(y, x)) \leftrightarrow x \in X) \rightarrow Mod_0^2(X))$$

Decision Problems [Courcelle:IC90]:

- Satisfiability for CMSO over trees is *decidable*
- Model-checking for CMSO over trees
 - combined complexity : *PSPACE-complete*
 - data complexity : *linear*

Beyond MSO: Counting MSO [Courcelle:IC90]

$Mod_i^j(X)$ holds if $card(\sigma(X)) \bmod i = j$

- “the number of edges in the tree is even”

$$\exists X \forall x (x \in X \wedge Mod_0^2(X))$$

- “the root has an even number of out-going edges”

$$\forall X (\forall x (\exists y \text{ root}(y) \wedge \text{edge}(y, x)) \leftrightarrow x \in X) \rightarrow Mod_0^2(X))$$

Decision Problems [Courcelle:IC90]:

- Satisfiability for CMSO over trees is *decidable*
- Model-checking for CMSO over trees
 - ▶ combined complexity : *PSPACE-complete*
 - ▶ data complexity : *linear*

Beyond MSO: Counting MSO [Courcelle:IC90]

$Mod_i^j(X)$ holds if $\text{card}(\sigma(X)) \bmod i = j$

- “the number of edges in the tree is even”

$$\exists X \forall x (x \in X \wedge Mod_0^2(X))$$

- “the root has an even number of out-going edges”

$$\forall X (\forall x (\exists y \text{ root}(y) \wedge \text{edge}(y, x)) \leftrightarrow x \in X) \rightarrow Mod_0^2(X))$$

Decision Problems [Courcelle:IC90]:

- Satisfiability for CMSO over trees is *decidable*
- Model-checking for CMSO over trees
 - ▶ combined complexity : *PSPACE-complete*
 - ▶ data complexity : *linear*

Beyond MSO: Presburger MSO [Seidl et al:PODS03]

Let p be a Presburger formula, ie a FO formula interpreted over $(\mathbb{N}, +, =)$ with u_{X_1}, \dots, u_{X_n} as free variables

x/p holds if $\langle u_{X_1} \mapsto \text{card}(S_1), \dots, u_{X_n} \mapsto \text{card}(S_n) \rangle \models p$
where $S_i = \sigma(X_i) \cap \text{children}(x)$ for all $1 \leq i \leq n$.

- each node has as many out-going edges labeled with a than with b

$\forall X_a, X_b \forall x$

$(\forall z (\exists y \text{label}_a(y, z) \leftrightarrow z \in X_a) \wedge (\exists y \text{label}_b(y, z) \leftrightarrow z \in X_b)) \rightarrow x/(u_{X_a} = u_{X_b})$

Decision Problems [Seidl et al:PODS03]:

- Satisfiability for PMSO over trees is *decidable*
- Model-checking for PMSO over trees
 - ▶ combined complexity : cf satisfiability for Presburger formulas
 - ▶ data complexity : *linear*

Beyond MSO: Presburger MSO [Seidl et al:PODS03]

Let p be a Presburger formula, ie a FO formula interpreted over $(\mathbb{N}, +, =)$ with u_{X_1}, \dots, u_{X_n} as free variables

x/p holds if $\langle u_{X_1} \mapsto \text{card}(S_1), \dots, u_{X_n} \mapsto \text{card}(S_n) \rangle \models p$
where $S_i = \sigma(X_i) \cap \text{children}(x)$ for all $1 \leq i \leq n$.

- each node has as many out-going edges labeled with a than with b

$\forall X_a, X_b \forall x$

$(\forall z (\exists y \text{label}_a(y, z) \leftrightarrow z \in X_a) \wedge (\exists y \text{label}_b(y, z) \leftrightarrow z \in X_b)) \rightarrow x/(u_{X_a} = u_{X_b})$

Decision Problems [Seidl et al:PODS03]:

- Satisfiability for PMSO over trees is *decidable*
- Model-checking for PMSO over trees
 - ▶ combined complexity : cf satisfiability for Presburger formulas
 - ▶ data complexity : *linear*

Beyond MSO: Presburger MSO [Seidl et al:PODS03]

Let p be a Presburger formula, ie a FO formula interpreted over $(\mathbb{N}, +, =)$ with u_{X_1}, \dots, u_{X_n} as free variables

x/p holds if $\langle u_{X_1} \mapsto \text{card}(S_1), \dots, u_{X_n} \mapsto \text{card}(S_n) \rangle \models p$
where $S_i = \sigma(X_i) \cap \text{children}(x)$ for all $1 \leq i \leq n$.

- each node has as many out-going edges labeled with a than with b

$\forall X_a, X_b \forall x$

$(\forall z (\exists y \text{label}_a(y, z) \leftrightarrow z \in X_a) \wedge (\exists y \text{label}_b(y, z) \leftrightarrow z \in X_b)) \rightarrow x/(u_{X_a} = u_{X_b})$

Decision Problems [Seidl et al:PODS03]:

- Satisfiability for PMSO over trees is *decidable*
- Model-checking for PMSO over trees
 - ▶ combined complexity : cf satisfiability for Presburger formulas
 - ▶ data complexity : *linear*

An Algebraic View of Trees

We consider operations over Tree_Λ , the set of trees with edges labeled over Λ

• $\mathbf{0}$ 

• for each a in Λ ,

$$a \left(\begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array} \right) = \begin{array}{c} \text{ } \\ | \\ \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array}$$


•


$$\begin{array}{c} \text{ } \\ | \\ \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array} \quad \Big| \quad \begin{array}{c} \text{ } \\ | \\ \text{ } \\ | \\ \text{ } \end{array} = \begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \\ \swarrow \quad \searrow \quad | \\ \text{ } \quad \text{ } \quad \text{ } \end{array}$$

NB The domain Tree_Λ is finitely generated by the operations $\mathbf{0}$, $(a)_{a \in \Lambda}$, $|$.

NB $|$ is associative and commutative and $\mathbf{0}$ is its neutral element.

An Algebraic View of Trees

We consider operations over Tree_Λ , the set of trees with edges labeled over Λ

• $\mathbf{0}$ 

• for each a in Λ ,

$$a \left(\begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array} \right) = \begin{array}{c} \text{ } \\ \downarrow a \\ \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array}$$

•

$$\begin{array}{c} \text{ } \\ \downarrow a \\ \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array} \mid \begin{array}{c} \text{ } \\ \downarrow b \\ \text{ } \\ \downarrow c \\ \text{ } \end{array} = \begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \\ \swarrow \quad \searrow \quad \downarrow c \\ \text{ } \quad \text{ } \quad \text{ } \end{array}$$

NB The domain Tree_Λ is finitely generated by the operations $\mathbf{0}, (a)_{a \in \Lambda}, |$.

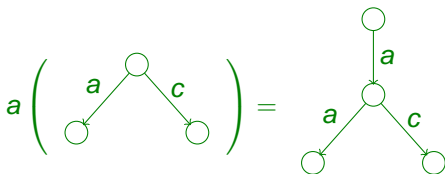
NB $|$ is associative and commutative and $\mathbf{0}$ is its neutral element.

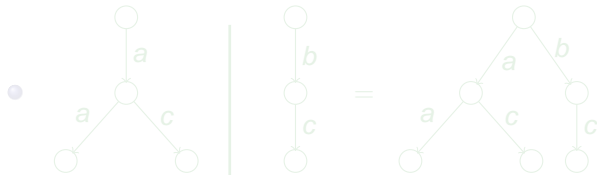
An Algebraic View of Trees

We consider operations over Tree_Λ , the set of trees with edges labeled over Λ

• $\mathbf{0}$ 

• for each a in Λ ,

$$a \left(\begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array} \right) = \begin{array}{c} \text{ } \\ | \\ \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array}$$


• 

NB The domain Tree_Λ is finitely generated by the operations $\mathbf{0}$, $(a)_{a \in \Lambda}$, $|$.

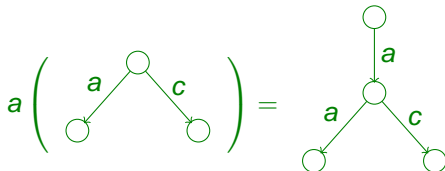
NB $|$ is associative and commutative and $\mathbf{0}$ is its neutral element.

An Algebraic View of Trees

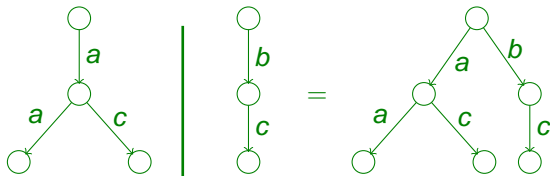
We consider operations over Tree_Λ , the set of trees with edges labeled over Λ

• $\mathbf{0}$ 

• for each a in Λ ,

$$a \left(\begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array} \right) = \begin{array}{c} \text{ } \\ \downarrow a \\ \text{ } \end{array} \begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array}$$


•


$$\begin{array}{c} \text{ } \\ \downarrow a \\ \text{ } \end{array} \begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array} \Big| \begin{array}{c} \text{ } \\ \downarrow b \\ \text{ } \\ \downarrow c \\ \text{ } \end{array} = \begin{array}{c} \text{ } \\ \downarrow b \\ \text{ } \end{array} \begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array}$$

NB The domain Tree_Λ is finitely generated by the operations $\mathbf{0}$, $(a)_{a \in \Lambda}$, $|$.

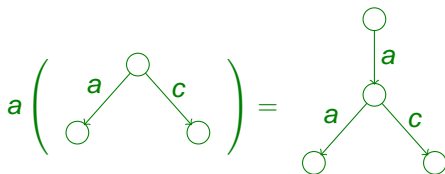
NB $|$ is associative and commutative and $\mathbf{0}$ is its neutral element.

An Algebraic View of Trees

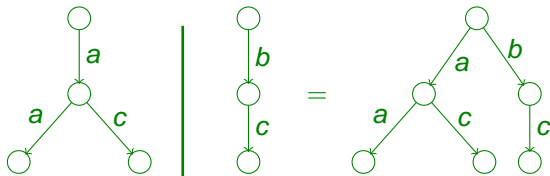
We consider operations over Tree_Λ , the set of trees with edges labeled over Λ

• $\mathbf{0}$ 

• for each a in Λ ,

$$a \left(\begin{array}{c} \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array} \right) = \begin{array}{c} \text{ } \\ \downarrow a \\ \text{ } \\ \swarrow \quad \searrow \\ \text{ } \quad \text{ } \end{array}$$


•



NB The domain Tree_Λ is finitely generated by the operations $\mathbf{0}$, $(a)_{a \in \Lambda}$, $|$.

NB $|$ is associative and commutative and $\mathbf{0}$ is its neutral element.

Algebraic Recognizability and CMSO

Definition (Mezei&Wright:IC67)

A set of trees L is **(algebraically) recognizable** if there exists

- a finite algebra \mathbb{A} over $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ whose domain is $D_{\mathbb{A}}$
- a homomorphism h from the tree algebra to \mathbb{A}
- a subset F of $D_{\mathbb{A}}$

such that $L = h^{-1}(F)$.

Algebraic Recognizability and CMSO

Definition (Mezei&Wright:IC67)

A set of trees L is **(algebraically) recognizable** if there exists

- a finite algebra \mathbb{A} over $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ whose domain is $D_{\mathbb{A}}$
- a homomorphism h from the tree algebra to \mathbb{A}
- a subset F of $D_{\mathbb{A}}$

such that $L = h^{-1}(F)$.

the number of edges in the tree is even: $h^{-1}(q_e)$

$$\begin{array}{l} \mathbf{0}^{\mathbb{A}} = q_e \\ a^{\mathbb{A}}(q_o) = q_e \quad a^{\mathbb{A}}(q_e) = q_o \\ b^{\mathbb{A}}(q_o) = q_e \quad b^{\mathbb{A}}(q_e) = q_o \\ q_o \stackrel{\mathbb{A}}{|} q_o = q_e \quad q_e \stackrel{\mathbb{A}}{|} q_e = q_e \\ q_o \stackrel{\mathbb{A}}{|} q_e = q_o \quad q_e \stackrel{\mathbb{A}}{|} q_o = q_o \\ q_e \stackrel{\mathbb{A}}{|} q_o = q_o \quad q_o \stackrel{\mathbb{A}}{|} q_e = q_o \end{array}$$

Algebraic Recognizability and CMSO

Definition (Mezei&Wright:IC67)

A set of trees L is **(algebraically) recognizable** if there exists

- a finite algebra \mathbb{A} over $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ whose domain is $D_{\mathbb{A}}$
- a homomorphism h from the tree algebra to \mathbb{A}
- a subset F of $D_{\mathbb{A}}$

such that $L = h^{-1}(F)$.

Theorem (Courcelle:IC90)

A set of trees is (algebraically) recognizable iff it is **CMSO-definable**.

Equationnality and PMSO

Definition (Mezei&Wright:IC67)

A set of trees L is **equationnal** if the value of some unknowns U_1 in the least solution over sets of trees of a system of equations of the form

$$U_1 = T_1$$

...

$$U_k = T_k$$

where the U_i 's are unknowns and the T_j 's terms built from the U_i 's, $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ and \cup .

\cup is interpreted as the union and the interpretation of $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ is extended canonically to sets.

Equationnality and PMSO

Definition (Mezei&Wright:IC67)

A set of trees L is **equationnal** if the value of some unknowns U_1 in the least solution over sets of trees of a system of equations of the form

$$U_1 = T_1$$

$$\dots$$
$$U_k = T_k$$

where the U_i 's are unknowns and the T_j 's terms built from the U_i 's, $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ and \cup .

\cup is interpreted as the union and the interpretation of $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ is extended canonically to sets.

each node has as many out-going edges labeled with a than with b

$$U = (U_a \mid U_b \mid U) \cup \mathbf{0}$$

$$U_a = a(U)$$

$$U_b = b(U)$$

Equationnality and PMSO

Definition (Mezei&Wright:IC67)

A set of trees L is **equationnal** if the value of some unknowns U_1 in the least solution over sets of trees of a system of equations of the form

$$U_1 = T_1$$

...

$$U_k = T_k$$

where the U_i 's are unknowns and the T_j 's terms built from the U_i 's, $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ and \cup .

\cup is interpreted as the union and the interpretation of $\{\mathbf{0}, (a)_{a \in \Lambda}, |\}\}$ is extended canonically to sets.

Theorem (Boneva&Talbot:RTA05)

*A set of trees is equationnal iff it is **PMSO-definable**.*

The Logic TQL^μ

TQL is based on the **ambients** [Cardelli-Ghelli:ESOP01]

TQL^μ is the quantifier-free fragment of TQL

Let $a \in \Lambda$ and countably many **recursion variables** (ξ, ξ', \dots)

Definition

A **formula** ϕ of the TQL^μ

\top	true	$\mathbf{0}$	void	} coreTQL
$\neg\phi$	negation	$a[\phi]$	extension	
$\phi \vee \phi$	disjunction	$\phi \mid \phi$	composition	
	ξ	recursion variable		
	$\mu\xi.\phi$	least fixed point		

The Logic TQL^μ

TQL is based on the **ambients** [Cardelli-Ghelli:ESOP01]

TQL^μ is the quantifier-free fragment of TQL

Let $a \in \Lambda$ and countably many **recursion variables** (ξ, ξ', \dots)

Definition

A **formula** ϕ of the TQL^μ

$\phi ::=$	\top	true	$\mathbf{0}$	void	} coreTQL
	$\neg\phi$	negation	$a[\phi]$	extension	
	$\phi \vee \phi$	disjunction	$\phi \mid \phi$	composition	
		ξ	recursion variable		
		$\mu\xi.\phi$	least fixed point		

The semantics of TQL^μ

The **interpretation** of TQL^μ is given by a mapping $\llbracket \cdot \rrbracket_\delta$ (parametrized by a valuation $\delta : \{\xi, \xi', \dots\} \rightarrow \wp(\text{Tree}_\Lambda)$) from formulas to sets of trees.

- $\llbracket \top \rrbracket_\delta = \text{Tree}_\Lambda$
- $\llbracket \neg\phi \rrbracket_\delta = \text{Tree}_\Lambda \setminus \llbracket \phi \rrbracket_\delta$
- $\llbracket \phi_1 \vee \phi_2 \rrbracket_\delta = \llbracket \phi_1 \rrbracket_\delta \cup \llbracket \phi_2 \rrbracket_\delta$
- $\llbracket \mathbf{0} \rrbracket_\delta = \{\mathbf{0}\}$
- $\llbracket a[\phi] \rrbracket_\delta = \{a(\tau) \mid \tau \in \llbracket \phi \rrbracket_\delta\}$

$\tau \in \llbracket a[\phi] \rrbracket_\delta$ iff $\tau =$



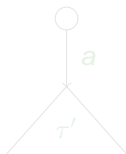
and $\tau' \in \llbracket \phi \rrbracket_\delta$.

The semantics of TQL^μ

The **interpretation** of TQL^μ is given by a mapping $\llbracket \cdot \rrbracket_\delta$ (parametrized by a valuation $\delta : \{\xi, \xi', \dots\} \rightarrow \wp(\text{Tree}_\Lambda)$) from formulas to sets of trees.

- $\llbracket \top \rrbracket_\delta = \text{Tree}_\Lambda$
- $\llbracket \neg\phi \rrbracket_\delta = \text{Tree}_\Lambda \setminus \llbracket \phi \rrbracket_\delta$
- $\llbracket \phi_1 \vee \phi_2 \rrbracket_\delta = \llbracket \phi_1 \rrbracket_\delta \cup \llbracket \phi_2 \rrbracket_\delta$
- $\llbracket \mathbf{0} \rrbracket_\delta = \{\mathbf{0}\}$
- $\llbracket a[\phi] \rrbracket_\delta = \{a(\tau) \mid \tau \in \llbracket \phi \rrbracket_\delta\}$

$\tau \in \llbracket a[\phi] \rrbracket_\delta$ iff $\tau =$



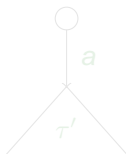
and $\tau' \in \llbracket \phi \rrbracket_\delta$.

The semantics of TQL^μ

The **interpretation** of TQL^μ is given by a mapping $\llbracket \cdot \rrbracket_\delta$ (parametrized by a valuation $\delta : \{\xi, \xi', \dots\} \rightarrow \wp(\text{Tree}_\Lambda)$) from formulas to sets of trees.

- $\llbracket \top \rrbracket_\delta = \text{Tree}_\Lambda$
- $\llbracket \neg\phi \rrbracket_\delta = \text{Tree}_\Lambda \setminus \llbracket \phi \rrbracket_\delta$
- $\llbracket \phi_1 \vee \phi_2 \rrbracket_\delta = \llbracket \phi_1 \rrbracket_\delta \cup \llbracket \phi_2 \rrbracket_\delta$
- $\llbracket \mathbf{0} \rrbracket_\delta = \{\mathbf{0}\}$
- $\llbracket \mathbf{a}[\phi] \rrbracket_\delta = \{\mathbf{a}(\tau) \mid \tau \in \llbracket \phi \rrbracket_\delta\}$

$\tau \in \llbracket \mathbf{a}[\phi] \rrbracket_\delta$ iff $\tau =$



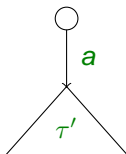
and $\tau' \in \llbracket \phi \rrbracket_\delta$.

The semantics of TQL^μ

The **interpretation** of TQL^μ is given by a mapping $\llbracket \cdot \rrbracket_\delta$ (parametrized by a valuation $\delta : \{\xi, \xi', \dots\} \rightarrow \wp(\text{Tree}_\Lambda)$) from formulas to sets of trees.

- $\llbracket \top \rrbracket_\delta = \text{Tree}_\Lambda$
- $\llbracket \neg\phi \rrbracket_\delta = \text{Tree}_\Lambda \setminus \llbracket \phi \rrbracket_\delta$
- $\llbracket \phi_1 \vee \phi_2 \rrbracket_\delta = \llbracket \phi_1 \rrbracket_\delta \cup \llbracket \phi_2 \rrbracket_\delta$
- $\llbracket \mathbf{0} \rrbracket_\delta = \{\mathbf{0}\}$
- $\llbracket \mathbf{a}[\phi] \rrbracket_\delta = \{\mathbf{a}(\tau) \mid \tau \in \llbracket \phi \rrbracket_\delta\}$

$\tau \in \llbracket \mathbf{a}[\phi] \rrbracket_\delta$ iff $\tau =$



and $\tau' \in \llbracket \phi \rrbracket_\delta$.

The semantics of TQL^μ (II)

- $[[\phi_1 \mid \phi_2]]_\delta = \{\tau_1 \mid \tau_2 \mid \tau_1 \in [[\phi_1]]_\delta, \tau_2 \in [[\phi_2]]_\delta\}$

$\tau \in [[\phi_1 \mid \phi_2]]_\delta$ iff $\tau =$

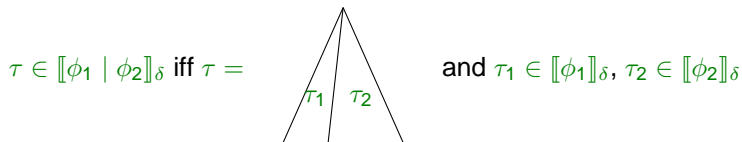


and $\tau_1 \in [[\phi_1]]_\delta, \tau_2 \in [[\phi_2]]_\delta$

- $[[\xi]]_\delta = \delta(\xi)$
- $[[\mu\xi.\phi]]_\delta = \bigcap \{S \subseteq \text{Tree}_\Lambda \mid [[\phi]]_{\delta[\xi \rightarrow S]} \subseteq S\}$

The semantics of TQL^μ (II)

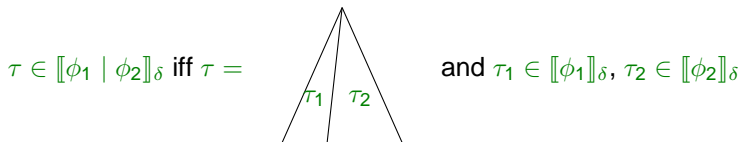
- $[[\phi_1 \mid \phi_2]]_\delta = \{\tau_1 \mid \tau_2 \mid \tau_1 \in [[\phi_1]]_\delta, \tau_2 \in [[\phi_2]]_\delta\}$



- $[[\xi]]_\delta = \delta(\xi)$
- $[[\mu\xi.\phi]]_\delta = \bigcap \{S \subseteq \text{Tree}_\Lambda \mid [[\phi]]_{\delta[\xi \rightarrow S]} \subseteq S\}$

The semantics of TQL^μ (II)

- $[[\phi_1 \mid \phi_2]]_\delta = \{\tau_1 \mid \tau_2 \mid \tau_1 \in [[\phi_1]]_\delta, \tau_2 \in [[\phi_2]]_\delta\}$



- $[[\xi]]_\delta = \delta(\xi)$
- $[[\mu\xi.\phi]]_\delta = \bigcap \{S \subseteq \mathbf{Tree}_\Lambda \mid [[\phi]]_{\delta[\xi \rightarrow S]} \subseteq S\}$

TQL^μ : examples

Over the set of labels $\{a, b\}$

- the root has at least two children :

$$(a[T] \vee b[T]) \mid (a[T] \vee b[T]) \mid T$$

- There is an edge labeled with a :

$$\mu\xi.(a[T] \mid T \vee (b[\xi] \mid T))$$

- there is a path from the root to a leaf labeled only by a :

$$\mu\xi.(a[0] \mid T \vee (a[\xi] \mid T))$$

TQL^μ : examples

Over the set of labels $\{a, b\}$

- the root has at least two children :

$$(a[T] \vee b[T]) \mid (a[T] \vee b[T]) \mid T$$

- There is an edge labeled with a :

$$\mu\xi.(a[T] \mid T \vee (b[\xi] \mid T))$$

- there is a path from the root to a leaf labeled only by a :

$$\mu\xi.(a[0] \mid T \vee (a[\xi] \mid T))$$

TQL^μ : examples

Over the set of labels $\{a, b\}$

- the root has at least two children :

$$(a[T] \vee b[T]) \mid (a[T] \vee b[T]) \mid T$$

- There is an edge labeled with a :

$$\mu\xi.(a[T] \mid T \vee (b[\xi] \mid T))$$

- there is a path from the root to a leaf labeled only by a :

$$\mu\xi.(a[0] \mid T \vee (a[\xi] \mid T))$$

TQL ^{μ} : examples beyond MSO

- the root has an even number of children:

$$\mu\xi.(\mathbf{0} \vee ((a[T] \vee b[T]) \mid (a[T] \vee b[T]) \mid \xi))$$

- each node has as many a children as b ones:

$$\mu\xi.(\mathbf{0} \vee a[\xi] \mid b[\xi] \mid \xi)$$

TQL^μ : examples beyond MSO

- the root has an even number of children:

$$\mu\xi.(\mathbf{0} \vee ((a[T] \vee b[T]) \mid (a[T] \vee b[T]) \mid \xi))$$

- each node has as many *a* children as *b* ones:

$$\mu\xi.(\mathbf{0} \vee a[\xi] \mid b[\xi] \mid \xi)$$

Model-checking TQL^μ

Definition (Model-checking)

For a closed formula ϕ and a tree τ , decide whether $\tau \in \llbracket \phi \rrbracket$.

Proposition (Boneva-Talbot:IFIP'TCS04)

*The combined complexity of **model-checking for TQL^μ** is in PSPACE.*

Proof.

Not so difficult but one has to be careful:

- the number of decomposition of a tree is exponential in its size.
- dealing with fixed points is always messy !!



Model-checking TQL^μ

Definition (Model-checking)

For a closed formula ϕ and a tree τ , decide whether $\tau \in \llbracket \phi \rrbracket$.

Proposition (Boneva-Talbot:IFIP'TCS04)

*The combined complexity of **model-checking for TQL^μ** is in PSPACE.*

Proof.

Not so difficult but one has to be careful:

- the number of decomposition of a tree is exponential in its size.
- dealing with fixed points is always messy !!



Model-checking TQL^μ

Proposition

The combined complexity of **model-checking for coreTQL** is PSPACE-hard.

Proof.

Encoding QBF (validity for Quantified Boolean Formulas)

Boolean valuations as trees: $v_1(t(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-free Boolean formulas as coreTQL formulas:

$$v_1 \wedge (v_2 \vee \neg v_3) \simeq v_1[t[\mathbf{0}]] \wedge (v_2[t[\mathbf{0}]] \vee v_3[f[\mathbf{0}]])$$

Starting from $v_1(t(\mathbf{0})) \mid v_1(f(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_2(f(\mathbf{0})) \mid v_3(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-prefixes: $\exists v_i.\psi \simeq v_i[\top] \mid \phi_\psi$ and $\forall v_i.\psi \simeq \neg(v_i[\top] \mid \neg\phi_\psi)$ □

Model-checking TQL^μ

Proposition

The combined complexity of **model-checking for coreTQL** is PSPACE-hard.

Proof.

Encoding QBF (validity for Quantified Boolean Formulas)

Boolean valuations as trees: $v_1(t(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-free Boolean formulas as coreTQL formulas:

$$v_1 \wedge (v_2 \vee \neg v_3) \simeq v_1[t[\mathbf{0}]] \wedge (v_2[t[\mathbf{0}]] \vee v_3[f[\mathbf{0}]])$$

Starting from $v_1(t(\mathbf{0})) \mid v_1(f(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_2(f(\mathbf{0})) \mid v_3(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-prefixes: $\exists v_i.\psi \simeq v_i[\mathbf{T}] \mid \phi_\psi$ and $\forall v_i.\psi \simeq \neg(v_i[\mathbf{T}] \mid \neg\phi_\psi)$ □

Model-checking TQL^μ

Proposition

The combined complexity of **model-checking for coreTQL** is PSPACE-hard.

Proof.

Encoding QBF (validity for Quantified Boolean Formulas)

Boolean valuations as trees: $v_1(t(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-free Boolean formulas as coreTQL formulas:

$$v_1 \wedge (v_2 \vee \neg v_3) \simeq v_1[t[\mathbf{0}]] \wedge (v_2[t[\mathbf{0}]] \vee v_3[f[\mathbf{0}]])$$

Starting from $v_1(t(\mathbf{0})) \mid v_1(f(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_2(f(\mathbf{0})) \mid v_3(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-prefixes: $\exists v_i.\psi \simeq v_i[\top] \mid \phi_\psi$ and $\forall v_i.\psi \simeq \neg(v_i[\top] \mid \neg\phi_\psi)$ □

Model-checking TQL^μ

Proposition

The combined complexity of **model-checking for coreTQL** is PSPACE-hard.

Proof.

Encoding QBF (validity for Quantified Boolean Formulas)

Boolean valuations as trees: $v_1(t(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-free Boolean formulas as coreTQL formulas:

$$v_1 \wedge (v_2 \vee \neg v_3) \simeq v_1[t[\mathbf{0}]] \wedge (v_2[t[\mathbf{0}]] \vee v_3[f[\mathbf{0}]])$$

Starting from $v_1(t(\mathbf{0})) \mid v_1(f(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_2(f(\mathbf{0})) \mid v_3(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-prefixes: $\exists v_i.\psi \simeq v_i[\mathbf{T}] \mid \phi_\psi$ and $\forall v_i.\psi \simeq \neg(v_i[\mathbf{T}] \mid \neg\phi_\psi)$ □

Model-checking TQL^μ

Proposition

The combined complexity of **model-checking for coreTQL** is PSPACE-hard.

Proof.

Encoding QBF (validity for Quantified Boolean Formulas)

Boolean valuations as trees: $v_1(t(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-free Boolean formulas as coreTQL formulas:

$$v_1 \wedge (v_2 \vee \neg v_3) \simeq v_1[t[\mathbf{0}]] \wedge (v_2[t[\mathbf{0}]] \vee v_3[f[\mathbf{0}]])$$

Starting from $v_1(t(\mathbf{0})) \mid v_1(f(\mathbf{0})) \mid v_2(t(\mathbf{0})) \mid v_2(f(\mathbf{0})) \mid v_3(t(\mathbf{0})) \mid v_3(f(\mathbf{0}))$

Quantifier-prefixes: $\exists v_i.\psi \simeq v_i[\top] \mid \phi_\psi$ and $\forall v_i.\psi \simeq \neg(v_i[\top] \mid \neg\phi_\psi)$ □

Satisfiability for the TQL Logic

Definition (Satisfiability)

For a closed formula ϕ , decide whether $\llbracket \phi \rrbracket$ is empty.

Proposition (Boneva-Talbot-Tison:LICS05)

The satisfiability problem is

- *decidable for coreTQL*

Proof.

- Follows from the encoding of coreTQL in MSO. See also [Dal Zilio et al:POPL04] and [Calcagno et al:TLDI03].



Satisfiability for the TQL Logic

Definition (Satisfiability)

For a closed formula ϕ , decide whether $\llbracket \phi \rrbracket$ is empty.

Proposition (Boneva-Talbot-Tison:LICS05)

The satisfiability problem is

- *decidable for coreTQL*
- *undecidable for TQL^μ*

Proof.

- Follows from the encoding of coreTQL in MSO. See also [Dal Zilio et al:POPL04] and [Calcagno et al::TLDI03].
- Reduction to emptiness of two-counters machines (following ideas from [Verma:PhD03]).



Satisfiability for the TQL Logic

Definition (Satisfiability)

For a closed formula ϕ , decide whether $\llbracket \phi \rrbracket$ is empty.

Proposition (Boneva-Talbot-Tison:LICS05)

The satisfiability problem is

- *decidable for coreTQL*
- *undecidable for TQL ^{μ}*

Proof.

- Follows from the encoding of coreTQL in MSO. See also [Dal Zilio et al:POPL04] and [Calcagno et al.:TLDI03].
- Reduction to emptiness of two-counters machines (following ideas from [Verma:PhD03]).



Tree Automata for MSO

Tree automata exists for extensions of MSO

- Counting MSO:
 - ▶ algebraic recognizability [Courcelle:IC90]:
 - ▶ tree automata based on recognizable sets of finite multisets
- Presburger MSO [Seidl-Schwentick-Muscholl:PODS2003]:
 - ▶ tree automata with transitions with Presburger constraints
 - ▶ Sheaves automata [Dal Zilio-Lugiez-Meyssonier:POPL04]
 - ▶ Rational multiset tree automata [Colcombet:ENTCS02]

[BonevaTalbot:RTA05]

A unified tree automata framework for MSO, CMSO, PMSO

- as for PMSO, transitions are based on arithmetical constraints
- a specific form of arithmetical constraints for each logic.

Tree Automata for MSO

Tree automata exists for extensions of MSO

- Counting MSO:
 - ▶ algebraic recognizability [Courcelle:IC90]:
 - ▶ tree automata based on recognizable sets of finite multisets
- Presburger MSO [Seidl-Schwentick-Muscholl:PODS2003]:
 - ▶ tree automata with transitions with Presburger constraints
 - ▶ Sheaves automata [Dal Zilio-Lugiez-Meyssonier:POPL04]
 - ▶ Rational multiset tree automata [Colcombet:ENTCS02]

[BonevaTalbot:RTA05]

A unified tree automata framework for MSO, CMSO, PMSO

- as for PMSO, transitions are based on arithmetical constraints
- a specific form of arithmetical constraints for each logic.

Tree Automata for MSO

Tree automata exists for extensions of MSO

- Counting MSO:
 - ▶ algebraic recognizability [Courcelle:IC90]:
 - ▶ tree automata based on recognizable sets of finite multisets
- Presburger MSO [Seidl-Schwentick-Muscholl:PODS2003]:
 - ▶ tree automata with transitions with Presburger constraints
 - ▶ Sheaves automata [Dal Zilio-Lugiez-Meyssonier:POPL04]
 - ▶ Rational multiset tree automata [Colcombet:ENTCS02]

[BonevaTalbot:RTA05]

A unified tree automata framework for MSO, CMSO, PMSO

- as for PMSO, transitions are based on arithmetical constraints
- a specific form of arithmetical constraints for each logic.

Tree Automata for MSO

Tree automata exists for extensions of MSO

- Counting MSO:
 - ▶ algebraic recognizability [Courcelle:IC90]:
 - ▶ tree automata based on recognizable sets of finite multisets
- Presburger MSO [Seidl-Schwentick-Muscholl:PODS2003]:
 - ▶ tree automata with transitions with Presburger constraints
 - ▶ Sheaves automata [Dal Zilio-Lugiez-Meyssonier:POPL04]
 - ▶ Rational multiset tree automata [Colcombet:ENTCS02]

[BonevaTalbot:RTA05]

A unified tree automata framework for MSO, CMSO, PMSO

- as for PMSO, transitions are based on arithmetical constraints
- a specific form of arithmetical constraints for each logic.

Tree Automata with arithmetical constraints

Definition

A **tree automata with arithmetical constraints** is given by:

- a finite set of states $Q = \{q_1, \dots, q_n\}$ (implicitly ordered)
- an acceptance condition $F \subseteq \mathbb{N}^n$
- the transitions Δ are given has a finite set of triples (q, β, N) where
 - ▶ $q \in Q$
 - ▶ $\beta \in \Lambda$
 - ▶ $N \subseteq \mathbb{N}^n$

Example:

- $Q = \{q_a, q_b, q_0\}$
- $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
- $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

Tree Automata with arithmetical constraints

Definition

A **tree automata with arithmetical constraints** is given by:

- a finite set of states $Q = \{q_1, \dots, q_n\}$ (implicitly ordered)
- an acceptance condition $F \subseteq \mathbb{N}^n$
- the transitions Δ are given has a finite set of triples (q, β, N) where
 - ▶ $q \in Q$
 - ▶ $\beta \in \Lambda$
 - ▶ $N \subseteq \mathbb{N}^n$

Example:

- $Q = \{q_a, q_b, q_0\}$
- $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
- $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

Tree Automata with arithmetical constraints

Definition

A **tree automata with arithmetical constraints** is given by:

- a finite set of states $Q = \{q_1, \dots, q_n\}$ (implicitly ordered)
- an acceptance condition $F \subseteq \mathbb{N}^n$
- the transitions Δ are given has a finite set of triples (q, β, N) where
 - ▶ $q \in Q$
 - ▶ $\beta \in \Lambda$
 - ▶ $N \subseteq \mathbb{N}^n$

Example:

- $Q = \{q_a, q_b, q_0\}$
- $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
- $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

Tree Automata with arithmetical constraints

Definition

A **tree automata with arithmetical constraints** is given by:

- a finite set of states $Q = \{q_1, \dots, q_n\}$ (implicitly ordered)
- an acceptance condition $F \subseteq \mathbb{N}^n$
- the transitions Δ are given has a finite set of triples (q, β, N) where
 - ▶ $q \in Q$
 - ▶ $\beta \in \Lambda$
 - ▶ $N \subseteq \mathbb{N}^n$

Example:

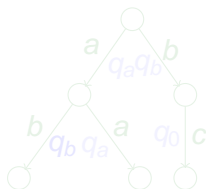
- $Q = \{q_a, q_b, q_0\}$
- $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
- $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

Tree Automata with arithmetical constraints: runs

Definition

A **run** r of the automaton maps edges to states from Q such that for all edges e labeled by $a \in \Lambda$, there exists

- $(r(e), a, N) \in \Delta$,
 - $(\#q_1, \dots, \#q_n) \in N$, where $\#q_i$ is the number of outgoing edges from the node targeted by e labeled by q_i in r
-
- $Q = \{q_a, q_b, q_0\}$
 - $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
 - $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

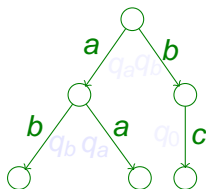


Tree Automata with arithmetical constraints: runs

Definition

A **run** r of the automaton maps edges to states from Q such that for all edges e labeled by $a \in \Lambda$, there exists

- $(r(e), a, N) \in \Delta$,
 - $(\#q_1, \dots, \#q_n) \in N$, where $\#q_i$ is the number of outgoing edges from the node targeted by e labeled by q_i in r
-
- $Q = \{q_a, q_b, q_0\}$
 - $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
 - $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

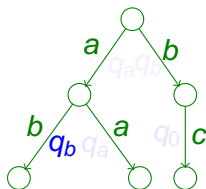


Tree Automata with arithmetical constraints: runs

Definition

A **run** r of the automaton maps edges to states from Q such that for all edges e labeled by $a \in \Lambda$, there exists

- $(r(e), a, N) \in \Delta$,
 - $(\#q_1, \dots, \#q_n) \in N$, where $\#q_i$ is the number of outgoing edges from the node targeted by e labeled by q_i in r
-
- $Q = \{q_a, q_b, q_0\}$
 - $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
 - $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

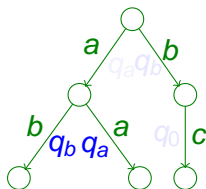


Tree Automata with arithmetical constraints: runs

Definition

A **run** r of the automaton maps edges to states from Q such that for all edges e labeled by $a \in \Lambda$, there exists

- $(r(e), a, N) \in \Delta$,
 - $(\#q_1, \dots, \#q_n) \in N$, where $\#q_i$ is the number of outgoing edges from the node targeted by e labeled by q_i in r
-
- $Q = \{q_a, q_b, q_0\}$
 - $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
 - $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

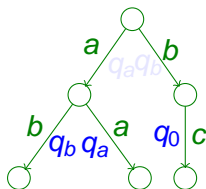


Tree Automata with arithmetical constraints: runs

Definition

A **run** r of the automaton maps edges to states from Q such that for all edges e labeled by $a \in \Lambda$, there exists

- $(r(e), a, N) \in \Delta$,
 - $(\#q_1, \dots, \#q_n) \in N$, where $\#q_i$ is the number of outgoing edges from the node targeted by e labeled by q_i in r
-
- $Q = \{q_a, q_b, q_0\}$
 - $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
 - $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

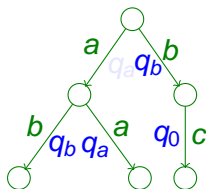


Tree Automata with arithmetical constraints: runs

Definition

A **run** r of the automaton maps edges to states from Q such that for all edges e labeled by $a \in \Lambda$, there exists

- $(r(e), a, N) \in \Delta$,
 - $(\#q_1, \dots, \#q_n) \in N$, where $\#q_i$ is the number of outgoing edges from the node targeted by e labeled by q_i in r
-
- $Q = \{q_a, q_b, q_0\}$
 - $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
 - $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$

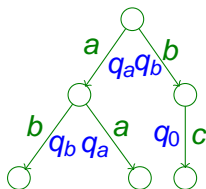


Tree Automata with arithmetical constraints: runs

Definition

A **run** r of the automaton maps edges to states from Q such that for all edges e labeled by $a \in \Lambda$, there exists

- $(r(e), a, N) \in \Delta$,
 - $(\#q_1, \dots, \#q_n) \in N$, where $\#q_i$ is the number of outgoing edges from the node targeted by e labeled by q_i in r
-
- $Q = \{q_a, q_b, q_0\}$
 - $F = \{(n, n, m) \mid n, m \in \mathbb{N}\}$
 - $N = \left\{ \begin{array}{l} (q_a, a, \{(n, n, m) \mid n, m \in \mathbb{N}\}), \\ (q_b, b, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \\ (q_0, c, \{(n, n, m) \mid n, m \in \mathbb{N}\}) \end{array} \right\}$



Classes of Tree Automata with arithmetical constraints

The acceptance condition F and any set N in triples (q, β, N) from Δ are

- a semilinear set, ie a finite union of linear sets, that is of the form

$$\{\vec{b} + \alpha_1 \vec{p}_1 + \dots + \alpha_k \vec{p}_k \mid \alpha_1, \dots, \alpha_k \in \mathbb{N}\} \text{ and } \vec{b}, \vec{p}_1, \dots, \vec{p}_k \in \mathbb{N}^n.$$

corresponds to **PMSO-definable** sets of trees

- a finite union of linear sets with periods collinear to unit vectors

corresponds to **CMSO-definable** sets of trees

- a finite union of linear sets with periods being unit vectors

corresponds to **MSO-definable** sets of trees

Classes of Tree Automata with arithmetical constraints

The acceptance condition F and any set N in triples (q, β, N) from Δ are

- a semilinear set, ie a finite union of linear sets, that is of the form

$$\{\vec{b} + \alpha_1 \vec{p}_1 + \dots + \alpha_k \vec{p}_k \mid \alpha_1, \dots, \alpha_k \in \mathbb{N}\} \text{ and } \vec{b}, \vec{p}_1, \dots, \vec{p}_k \in \mathbb{N}^n.$$

corresponds to **PMSO-definable** sets of trees

- a finite union of linear sets with periods collinear to unit vectors
corresponds to **CMSO-definable** sets of trees
- a finite union of linear sets with periods being unit vectors
corresponds to **MSO-definable** sets of trees

Classes of Tree Automata with arithmetical constraints

The acceptance condition F and any set N in triples (q, β, N) from Δ are

- a semilinear set, ie a finite union of linear sets, that is of the form

$$\{\vec{b} + \alpha_1 \vec{p}_1 + \dots + \alpha_k \vec{p}_k \mid \alpha_1, \dots, \alpha_k \in \mathbb{N}\} \text{ and } \vec{b}, \vec{p}_1, \dots, \vec{p}_k \in \mathbb{N}^n.$$

corresponds to **PMSO-definable** sets of trees

- a finite union of linear sets with periods collinear to unit vectors

corresponds to **CMSO-definable** sets of trees

- a finite union of linear sets with periods being unit vectors

corresponds to **MSO-definable** sets of trees

Classes of Tree Automata with arithmetical constraints

The acceptance condition F and any set N in triples (q, β, N) from Δ are

- a semilinear set, ie a finite union of linear sets, that is of the form

$$\{\vec{b} + \alpha_1 \vec{p}_1 + \dots + \alpha_k \vec{p}_k \mid \alpha_1, \dots, \alpha_k \in \mathbb{N}\} \text{ and } \vec{b}, \vec{p}_1, \dots, \vec{p}_k \in \mathbb{N}^n.$$

corresponds to **PMSO-definable** sets of trees

- a finite union of linear sets with periods collinear to unit vectors

corresponds to **CMSO-definable** sets of trees

- a finite union of linear sets with periods being unit vectors

corresponds to **MSO-definable** sets of trees

Tree Automata with arithmetical constraints and TQL^μ

From a closed TQL^μ formula φ , we extract a tree automaton \mathcal{A} with arithmetical constraints:

- states are built syntactically from the formula φ
- the sets N of triples from Δ are defined by a system of fixed point equations interpreted over $\wp(\mathbb{N}^n)$.

and such that the accepted language of \mathcal{A} is the set of models of the formula φ .

Emptiness is **undecidable** for those automata (cf. undecidability of TQL^μ) but automata can be useful anyway.

Tree Automata with arithmetical constraints and TQL^μ

From a closed TQL^μ formula φ , we extract a tree automaton \mathcal{A} with arithmetical constraints:

- states are built syntactically from the formula φ
- the sets N of triples from Δ are defined by a system of fixed point equations interpreted over $\wp(\mathbb{N}^n)$.

and such that the accepted language of \mathcal{A} is the set of models of the formula φ .

Emptiness is **undecidable** for those automata (cf. undecidability of TQL^μ) but automata can be useful anyway.

Tree Automata with arithmetical constraints and TQL^μ

Lemma

(Non-uniform) membership can be decided in PTIME for the automata obtained from TQL^μ formulas.

as an immediate consequence

Proposition

The data complexity of model-checking for TQL^μ is in PTIME.

Tree Automata with arithmetical constraints and TQL^μ

Lemma

(Non-uniform) membership can be decided in PTIME for the automata obtained from TQL^μ formulas.

as an immediate consequence

Proposition

*The data complexity of **model-checking for TQL^μ** is in PTIME.*

Tree Automata with arithmetical constraints and TQL^μ

[Boneva-Talbot-Tison:LICS05]

Thanks to our construction of tree automata, we identify syntactic restrictions over TQL^μ such that the fragments correspond to MSO and PMSO respectively.

- **guarded** TQL^μ whose expressiveness coincides with MSO
- **positively guarded** TQL^μ whose expressiveness coincides with PMSO

Moreover, by means of tree automata, all translations are effective.

Tree Automata with arithmetical constraints and TQL^μ

[Boneva-Talbot-Tison:LICS05]

Thanks to our construction of tree automata, we identify syntactic restrictions over TQL^μ such that the fragments correspond to MSO and PMSO respectively.

- **guarded TQL^μ** whose expressiveness coincides with MSO
- **positively guarded TQL^μ** whose expressiveness coincides with PMSO

Moreover, by means of tree automata, all translations are effective.

Tree Automata with arithmetical constraints and TQL^μ

[Boneva-Talbot-Tison:LICS05]

Thanks to our construction of tree automata, we identify syntactic restrictions over TQL^μ such that the fragments correspond to MSO and PMSO respectively.

- **guarded TQL^μ** whose expressiveness coincides with MSO
- **positively guarded TQL^μ** whose expressiveness coincides with PMSO

Moreover, by means of tree automata, all translations are effective.

Tree Automata with arithmetical constraints and TQL^μ

[Boneva-Talbot-Tison:LICS05]

Thanks to our construction of tree automata, we identify syntactic restrictions over TQL^μ such that the fragments correspond to MSO and PMSO respectively.

- **guarded TQL^μ** whose expressiveness coincides with MSO
- **positively guarded TQL^μ** whose expressiveness coincides with PMSO

Moreover, by means of tree automata, all translations are effective.

The Study of TQL: a (partial) summary

TQL ^μ	
MC (data)	PTIME
Sat	undecidable

positively guarded TQL ^μ	
MC (data)	linear
Sat	decidable

= PMSO

guarded TQL ^μ	
MC (data)	linear
Sat	decidable

= MSO

coreTQL	
MC (data)	linear
Sat	decidable