AUTOMATA ON INFINITE TREES WITH EQUALITY AND DISEQUALITY CONSTRAINTS BETWEEN SIBLINGS

Arnaud Carayol ¹  Christof Löding ²  Olivier Serre ³

¹LIGM (Université Paris Est & CNRS)
²RWTH Aachen
³IRIF (Université Paris Diderot – Paris 7 & CNRS)

**Non-deterministic Parity Tree Automata**

**Non-deterministic parity tree automata:**
\[ A = \langle Q, A, \Delta, q_{in}, \text{Col} \rangle \]

- **\( Q \):** control states
- **\( A \):** labels alphabet
- **\( \Delta \subseteq Q \times A \times Q \times Q \):** transition relation
- **\( q_{in} \):** initial state
- **\( \text{Col}: Q \to \mathbb{N} \):** colouring function

**Run on an \( A \)-labeled (infinite binary) tree \( t \):** \( Q \)-labelling of \( t \) consistent with \( \Delta \)

\[ \Delta = \{ \cdots (q_{in}, a, p, p) (p, b, q, p)(p, b, p, p) \cdots \} \]

A branch is **accepting** iff the smallest colour infinitely often visited is even

A run is **accepting** iff all its branches are accepting

A tree is **accepted** iff there is an accepting run over it.
**Tree Automata: Example**

\[ A = \{ a, b \} \]

\[ Q = \{ q_1, q_2, q_3 \} \]

<table>
<thead>
<tr>
<th>State Transition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_1 A \rightarrow (q_1, q_3)</td>
<td>q_2 A \rightarrow (q_2, q_3)</td>
</tr>
<tr>
<td>q_1 A \rightarrow (q_2, q_3)</td>
<td>q_2 b \rightarrow (q_1, q_3)</td>
</tr>
<tr>
<td>q_1 b \rightarrow (q_1, q_3)</td>
<td>q_3 b \rightarrow (q_3, q_3)</td>
</tr>
<tr>
<td>q_3 a \rightarrow (q_3, q_3)</td>
<td></td>
</tr>
</tbody>
</table>

Initial state: q_1

F = \{ q_2, q_3 \}
Tree Automata: Example

$A = \{a, b\}$

$Q = \{q_1, q_2, q_3\}$

Initial state: $q_1$

$F = \{q_2, q_3\}$
Tree Automata: Example

\[ A = \{ a, b \} \]

\[ Q = \{ q_1, q_2, q_3 \} \]

\[ F = \{ q_2, q_3 \} \]

A branch is accepting if it has infinitely many occurrences of a state from \( F \) (Büchi).

A run is accepting if all its branches are accepting (\( \forall \)).

A tree is accepted if there exists an accepting run (9).
Tree Automata: Example

\[ A = \{ a, b \} \]

\[ Q = \{ q_1, q_2, q_3 \} \]

\[ F = \{ q_2, q_3 \} \]

Initial state: \( q_1 \)

A branch is accepting if it has infinitely many occurrences of a state from \( F \) (Büchi).

A run is accepting if all its branches are accepting (8).

A tree is accepted if there exists an accepting run (9).
**Tree Automata: Example**

$A = \{a, b\}$

$Q = \{q_1, q_2, q_3\}$

A branch is accepting if it has infinitely many occurrences of a state from $F$ (Büchi).

A run is accepting if all its branches are accepting ($\forall$).

A tree is accepted if there exists an accepting run ($\exists$).

Initial state: $q_1$

$F = \{q_2, q_3\}$
**Tree Automata: Example**

\[ A = \{a, b\} \]

\[ Q = \{q_1, q_2, q_3\} \]

\[ \begin{align*}
q_1 & \xrightarrow{A} (q_1, q_3) & q_2 & \xrightarrow{A} (q_2, q_3) \\
q_1 & \xrightarrow{A} (q_2, q_3) & q_2 & \xrightarrow{b} (q_1, q_3) \\
q_1 & \xrightarrow{b} (q_1, q_3) & q_3 & \xrightarrow{b} (q_3, q_3) \\
q_3 & \xrightarrow{a} (q_3, q_3)
\end{align*} \]

initial state: \( q_1 \)

\[ F = \{q_2, q_3\} \]
**Tree Automata: Example**

\[ A = \{ a, b \} \]

\[ Q = \{ q_1, q_2, q_3 \} \]

\[ F = \{ q_2, q_3 \} \]

\[ \begin{align*}
  q_1 & \xrightarrow{\mathcal{A}} (q_1, q_3) \\
  q_2 & \xrightarrow{\mathcal{A}} (q_2, q_3) \\
  q_1 & \xrightarrow{b} (q_1, q_3) \\
  q_2 & \xrightarrow{b} (q_1, q_3) \\
  q_3 & \xrightarrow{a} (q_3, q_3) \\
\end{align*} \]
Tree Automata: Example

$A = \{ a, b \}$

$Q = \{ q_1, q_2, q_3 \}$

$A$

$q_1 \xrightarrow{A} (q_1, q_3)$
$q_2 \xrightarrow{A} (q_2, q_3)$
$q_1 \xrightarrow{b} (q_1, q_3)$
$q_2 \xrightarrow{b} (q_1, q_3)$
$q_3 \xrightarrow{a} (q_3, q_3)$
$q_3 \xrightarrow{b} (q_3, q_3)$

initial state: $q_1$

$F = \{ q_2, q_3 \}$
Tree Automata: Example

$A = \{ a, b \}$

$Q = \{ q_1, q_2, q_3 \}$

A branch is accepting if it has infinitely many occurrences of a state from $F$ (Büchi).

A run is accepting if all its branches are accepting ($\forall$).

A tree is accepted if there exists an accepting run ($\exists$).

initial state: $q_1$

$F = \{ q_2, q_3 \}$
Tree Automata: Example

$A = \{a, b\}$

$Q = \{q_1, q_2, q_3\}$

$F = \{q_2, q_3\}$

A branch is accepting if it has infinitely many occurrences of a state from $F$ (Büchi).

A run is accepting if all its branches are accepting ($\forall$).

A tree is accepted if there exists an accepting run ($\exists$).
Tree Automata: Example

$A = \{a, b\}$

$Q = \{q_1, q_2, q_3\}$

$A$

$q_1 \xrightarrow{A} (q_1, q_3)$
$q_2 \xrightarrow{A} (q_2, q_3)$
$q_1 \xrightarrow{b} (q_1, q_3)$
$q_2 \xrightarrow{b} (q_1, q_3)$
$q_3 \xrightarrow{a} (q_3, q_3)$

initial state: $q_1$

$F = \{q_2, q_3\}$
Tree Automata: Example

\[ A = \{a, b\} \]

\[ Q = \{q_1, q_2, q_3\} \]

Initial state: \(q_1\)

Final states: \(\{q_2, q_3\}\)
**Tree Automata: Example**

\[ A = \{a, b\} \]

\[ Q = \{q_1, q_2, q_3\} \]

- \[ q_1 \xrightarrow{A} (q_1, q_3) \]
- \[ q_2 \xrightarrow{A} (q_2, q_3) \]
- \[ q_1 \xrightarrow{b} (q_1, q_3) \]
- \[ q_2 \xrightarrow{b} (q_1, q_3) \]
- \[ q_3 \xrightarrow{a} (q_3, q_3) \]

Initial state: \( q_1 \)

\[ F = \{q_2, q_3\} \]
Tree Automata: Example

$A = \{a, b\}$

$Q = \{q_1, q_2, q_3\}$

$\begin{align*}
q_1 & \xrightarrow{A} (q_1, q_3) \\
q_1 & \xrightarrow{b} (q_1, q_3) \\
q_2 & \xrightarrow{A} (q_2, q_3) \\
q_2 & \xrightarrow{b} (q_1, q_3) \\
q_3 & \xrightarrow{a} (q_3, q_3) \\
q_3 & \xrightarrow{b} (q_3, q_3)
\end{align*}$

initial state: $q_1$

$F = \{q_2, q_3\}$

A branch is accepting if it has infinitely many occurrences of a state from $F$ (Büchi).

A run is accepting if all its branches are accepting ($\forall$).

A tree is accepted if there exists an accepting run ($\exists$).
A branch is accepting if it has infinitely many occurrences of a state from $F$ (Büchi).

A run is accepting if all its branches are accepting ($\forall$).

A tree is accepted if there exists an accepting run ($\exists$).
Tree Automata: Example

\[ A = \{ a, b \} \]

\[ Q = \{ q_1, q_2, q_3 \} \]

\[ \begin{align*}
    q_1 & \xrightarrow{A} (q_1, q_3) \\
    q_1 & \xrightarrow{A} (q_2, q_3) \\
    q_1 & \xrightarrow{b} (q_1, q_3) \\
    q_3 & \xrightarrow{a} (q_3, q_3) \\
    q_2 & \xrightarrow{A} (q_2, q_3) \\
    q_2 & \xrightarrow{b} (q_1, q_3) \\
    q_3 & \xrightarrow{b} (q_3, q_3)
\end{align*} \]

initial state: \( q_1 \)

\[ F = \{ q_2, q_3 \} \]

A branch is accepting if it has infinitely many occurrences of a state from \( F \) (Büchi).

A run is accepting if all its branches are accepting (\( \forall \)).

A tree is accepted if there exists an accepting run (\( \exists \)).
**Tree Automata: Example**

Let $A = \{a, b\}$.

$$Q = \{q_1, q_2, q_3\}$$

$$\begin{align*}
q_1 \xrightarrow{A} (q_1, q_3) & \quad q_2 \xrightarrow{A} (q_2, q_3) \\
q_1 \xrightarrow{A} (q_2, q_3) & \quad q_2 \xrightarrow{b} (q_1, q_3) \\
q_1 \xrightarrow{b} (q_1, q_3) & \quad q_3 \xrightarrow{b} (q_3, q_3) \\
q_3 \xrightarrow{a} (q_3, q_3) &
\end{align*}$$

Initial state: $q_1$

$$F = \{q_2, q_3\}$$

A branch is accepting if it has infinitely many occurrences of a state from $F$ (Büchi).

A run is accepting if all its branches are accepting ($\forall$).

A tree is accepted if there exists an accepting run ($\exists$).
A subset $L$ of trees is **regular** if there exists some non-deterministic parity tree automaton $A$ such that $L = L(A)$.

Regular trees languages have many nice properties, among other:

- Coincide with MSO definable languages (hence, expressive).
- Form an effective Boolean algebra.
- Decidable emptiness and cardinality problem.

Whether the class can be extended while preserving (most of) its good properties is a challenging problem.
A subset $L$ of trees is **regular** if there exists some non-deterministic parity tree automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.

Regular trees languages have many nice properties, among other:

- Coincide with MSO definable languages (hence, expressive).
- Form an effective Boolean algebra.
- Decidable emptiness and cardinality problem.

Whether the class can be extended while preserving (most of) its good properties is a challenging problem.

We address this question by considering automata that can check equality between siblings.
Tree Automata With Constraints

Main idea: works as usual tree automata except that transitions can be guarded by an equality/disequality requirement on siblings.
**Tree Automata With Constraints**

**Main idea:** works as usual tree automata except that transitions can be guarded by an equality/disequality requirement on siblings.

**Formally:** With any $A$-labelled tree $t$ associate an $A \times \{=, \neq\}$ tree $t^?\bar{=}$ by annotating every node $u$ in $t$ by an extra information regarding on whether the left and the right subtrees rooted at $u$ are equal or not. More formally, for every $u \in \{0, 1\}^*$,

$$t^?\bar{=} (u) = \begin{cases}  (t(u), =) & \text{if } t[u0] = t[u1] \\  (t(u), \neq) & \text{if } t[u0] \neq t[u1] \end{cases}$$
Tree Automata With Constraints

**Main idea:** works as usual tree automata except that transitions can be guarded by an equality/disequality requirement on siblings.

**Formally:** With any $A$-labelled tree $t$ associate an $A \times \{=, \neq\}$ tree $t^\equiv$ by annotating every node $u$ in $t$ by an extra information regarding on whether the left and the right subtrees rooted at $u$ are equal or not. More formally, for every $u \in \{0, 1\}^*$,

$$t^\equiv(u) = \begin{cases} 
(t(u), =) & \text{if } t[u0] = t[u1] \\
(t(u), \neq) & \text{if } t[u0] \neq t[u1]
\end{cases}$$

An automaton $A$ with constraints over alphabet $A$ is an automaton over alphabet $A \times \{=, \neq\}$ and one lets

$$L_{con}(A) = \{ t \mid t^\equiv \in L(A) \}$$
Properties of Languages Accepted by Automata with Constraints

\( \text{REG} = \)?: class of languages recognised by automata with constraints.

**Theorem.** The class \( \text{REG} = \) is an effective Boolean algebra.

**Conjecture.** The class \( \text{REG} = \) is not closed under projection.
Properties of Languages Accepted by Automata with Constraints

$\text{REG}^\equiv$: class of languages recognised by automata with constraints.

**Theorem.** The class $\text{REG}^\equiv$ is an effective Boolean algebra.

**Conjecture.** The class $\text{REG}^\equiv$ is not closed under projection.

It captures natural properties beyond MSO like “the tree satisfies $\varphi$ and $\forall x$ if the subtree at $x$ satisfies $\varphi_1$, then its two subtrees are different/equal, and satisfy $\varphi_2$.”
Properties of Languages Accepted by Automata with Constraints

\( \text{REG}^? \): class of languages recognised by automata with constraints.

**Theorem.** The class \( \text{REG}^? \) is an effective Boolean algebra.

**Conjecture.** The class \( \text{REG}^? \) is not closed under projection.

It captures natural properties beyond MSO like “the tree satisfies \( \varphi \) and \( \forall x \) if the subtree at \( x \) satisfies \( \varphi_1 \), then its two subtrees are different/equal, and satisfy \( \varphi_2 \).”

**Proposition.** Let \( A \) be an automaton with constraints and let \( t \) be a regular tree. Then one can decide whether \( t \in L^{\text{con}}(A) \).
The Cardinality Problem

The **cardinality profile** \( \kappa_A \) of \( A \), is a mapping that assigns to each state \( q \) of \( A \) the cardinality of \( L^{con}(A_q) \).

**Proposition.** Let \( \mathbb{N}_0 \) be the cardinality of the set of natural numbers, and \( 2^{\mathbb{N}_0} \) the cardinality of the set of the real numbers. Then

\[
\kappa_A : Q \rightarrow \mathbb{N} \cup \{ \mathbb{N}_0, 2^{\mathbb{N}_0} \}
\]
The Cardinality Problem

The **cardinality profile** $\kappa_A$ of $A$, is a mapping that assigns to each state $q$ of $A$ the cardinality of $L^{con}(A_q)$.

**Proposition.** Let $\mathbb{N}_0$ be the cardinality of the set of natural numbers, and $2^{\mathbb{N}_0}$ the cardinality of the set of the real numbers. Then

$$\kappa_A : Q \rightarrow \mathbb{N} \cup \{\mathbb{N}_0, 2^{\mathbb{N}_0}\}$$

For **regular tree languages** it is known from [Niwinski’91] that one can compute the cardinality profile.
**The Cardinality Problem**

The **cardinality profile** $\kappa_A$ of $A$, is a mapping that assigns to each state $q$ of $A$ the cardinality of $L_{con}(A_q)$.

**Proposition.** Let $\mathbb{N}_0$ be the cardinality of the set of natural numbers, and $2^{\mathbb{N}_0}$ the cardinality of the set of the real numbers. Then

$$\kappa_A : Q \rightarrow \mathbb{N} \cup \{\mathbb{N}_0, 2^{\mathbb{N}_0}\}$$

For **regular tree languages** it is known from [Niwinski’91] that one can compute the cardinality profile.

Our main result is the following:

**Theorem.** Let $A$ be a parity tree automaton with constraints. Then, one can compute its cardinality profile.
Some Tools

First, get rids of equalities:

**Theorem.** Let $\mathcal{A}$ be an automaton with equality and disequality constraints. Then one can build an automaton $\mathcal{B}$ with **disequality everywhere** and s.t. $L^{\text{con}}(\mathcal{A})$ and $L^{\text{con}}(\mathcal{B})$ have the same cardinality.

Second, over-approximate the language $L^{\text{con}}(\mathcal{A}_{\text{q}})$ by $L(\mathcal{B}_{\text{q}})$ the language accepted by forgetting the constraints and use the results of [Niwinski’91].
Some Tools

First, get rids of equalities:

**Theorem.** Let $\mathcal{A}$ be an automaton with equality and disequality constraints. Then one can build an automaton $\mathcal{B}$ with **disequality everywhere** and s.t. $L^{\text{con}}(\mathcal{A})$ and $L^{\text{con}}(\mathcal{B})$ have the same cardinality.

Second, over-approximate the language $L^{\text{con}}(\mathcal{A}_q)$ by $L(\hat{\mathcal{A}})$ the language accepted by forgetting the constraints and use the results of [Niwinski’91] on it.
Example

Let $t_a/t_b$ be defined by $t_a(\varepsilon) = a$, $t_b(\varepsilon) = b$, $t_a(u0) = t_b(u0) = a$ and $t_a(u1) = t_b(u1) = b$ for any $u \in \{0, 1\}^*$. 
**Example**

Let \( t_a / t_b \) be defined by \( t_a(\varepsilon) = a, \ t_b(\varepsilon) = b, \ t_a(u0) = t_b(u0) = a \) and \( t_a(u1) = t_b(u1) = b \) for any \( u \in \{0, 1\}^* \).

Let \( A \) be the safety automaton (\( \{ q_{in}, q_b \}, \{ (a, \neq), (b, \neq) \}, q_{in}, \Delta_A \) where \( \Delta = \{ (q_{in}, (a, \neq), q_{in}, t_b), (q_{in}, (a, \neq), q_b, t_b), (q_b, (b, \neq), t_a, t_b) \} \).

Then, \( |L(\hat{A})| = \aleph_0 \). But, \( L^{con}(A) = \{ t_a \} \).
If $L(\hat{A})$ is countable, then it has a special shape. Namely there is a regular language of finite trees $L(B)$ such that the trees in $L(\hat{A})$ are exactly those obtained from a tree in $L(B)$ by replacing every leaf by a regular tree uniquely determined by the leaf label.
If $L(\hat{A})$ is countable, then it has a special shape. Namely there is a regular language of finite trees $L(B)$ such that the trees in $L(\hat{A})$ are exactly those obtained from a tree in $L(B)$ by replacing every leaf by a regular tree uniquely determined by the leaf label.
If \( L(\hat{A}) \) is countable, then it has a special shape. Namely there is a regular language of **finite** trees \( L(B) \) such that the trees in \( L(\hat{A}) \) are exactly those obtained from a tree in \( L(B) \) by replacing every leaf by a regular tree uniquely determined by the leaf label.

Let \( t_a/t_b \) be defined by \( t_a(\varepsilon) = a, \ t_b(\varepsilon) = b, \ t_a(u0) = t_b(u0) = a \) and \( t_a(u1) = t_b(u1) = b \) for any \( u \in \{0, 1\}^* \).

Let \( A \) be the safety automaton \( (\{q_{in}, q_b\}, \{(a, \neq), (b, \neq)\}, q_{in}, \Delta_A) \) where \( \Delta = \{(q_{in}, (a, \neq), q_{in}, t_b), (q_{in}, (a, \neq), q_b, t_b), (q_b, (b, \neq), t_a, t_b)\} \).
If $L(\widehat{A})$ is countable, then it has a special shape. Namely there is a regular language of \textbf{finite} trees $L(B)$ such that the trees in $L(\widehat{A})$ are exactly those obtained from a tree in $L(B)$ by replacing every leaf by a regular tree uniquely determined by the leaf label.

Let $t_a/t_b$ be defined by $t_a(\varepsilon) = a$, $t_b(\varepsilon) = b$, $t_a(u0) = t_b(u0) = a$ and $t_a(u1) = t_b(u1) = b$ for any $u \in \{0, 1\}^*$.

Let $A$ be the safety automaton $(\{q_{in}, q_b\}, \{(a, \neq), (b, \neq)\}, q_{in}, \Delta_A)$ where \[\Delta = \{(q_{in}, (a, \neq), q_{in}, t_b), (q_{in}, (a, \neq), q_b, t_b), (q_b, (b, \neq), t_a, t_b)\}.\]

Then $L(B)$ is (where $a \mapsto t_a$ and $b \mapsto t_b$):

\[
\begin{array}{c}
  a \\
  \quad \quad \quad a \\
  \quad \quad \quad \quad \quad \quad \quad \quad a \\
  \quad \quad \quad \quad \quad \quad \quad \quad \quad b \\
  \quad \quad \quad \quad \quad \quad b \\
  \quad \quad \quad b \\
  \quad b \\
\end{array}
\]
If $L(\hat{A})$ is countable, then it has a special shape. Namely there is a regular language of finite trees $L(B)$ such that the trees in $L(\hat{A})$ are exactly those obtained from a tree in $L(B)$ by replacing every leaf by a regular tree uniquely determined by the leaf label.

**Roadmap to compute the cardinality of $L(A)$ when $L(\hat{A})$ is countable:**

- Safely assume that $A$ has disequality everywhere.
- Built from $B$ an automaton on finite trees with constraints $C$ such that $L^{con}(A)$ and $L^{con}(C)$ have the same cardinal.
- Use the results from [Bogaert&Tison’02] to compute the cardinal of $L^{con}(C)$. 

**Countable Unconstrained Languages (2/2)**
Algorithm to Compute the Cardinality Profile

Input: Tree automaton with disequality constraints everywhere \( A \)

Data Structure:
- Set \( S \leftarrow Q \) the states of \( A \)
- Automaton \( B \leftarrow A \)
- Function \( \kappa : Q \rightarrow \mathbb{N} \cup \{ \aleph_0, 2^{\aleph_0} \}; \kappa(q) \leftarrow 2^{\aleph_0} \) for all \( q \)

Code:

1: while \( \exists q \in S \) s.t. \( |L(B_q)| \leq \aleph_0 \) do
2: \( \kappa(q) \leftarrow |L^{con}(B_q)| \)
3: if \( \kappa(q) = 0 \) then
4: \( B \leftarrow B_{q\rightarrow\emptyset} \)
5: else if \( \kappa(q) < \aleph_0 \) then
6: Let \( L^{con}(B_q) = \{ t_1, \ldots, t_n \} \)
7: \( B \leftarrow B_{q\rightarrow t_1,\ldots,t_n} \)
8: end if
9: \( S \leftarrow S \setminus \{ q \} \)
10: end while
11: return \( \kappa \)
Example of Execution

Recall that we defined $t_a/t_b$ by $t_a(\varepsilon) = a$, $t_b(\varepsilon) = b$, $t_a(u0) = t_b(u0) = a$ and $t_a(u1) = t_b(u1) = b$ for any $u \in \{0, 1\}^*$.

And $\mathcal{A}$ as the safety automaton $\langle \{q_{in}, q_b\}, \{(a, \neq), (b, \neq)\}, q_{in}, \Delta_\mathcal{A} \rangle$ where $\Delta = \{(q_{in}, (a, \neq), q_{in}, t_b), (q_{in}, (a, \neq), q_b, t_b), (q_b, (b, \neq), t_a, t_b)\}$.

$|L(\widehat{\mathcal{A}})| = \aleph_0$ but $L^{con}(\mathcal{A}) = \{t_a\}$. 
Example of Execution

Recall that we defined $t_a / t_b$ by $t_a(\varepsilon) = a$, $t_b(\varepsilon) = b$, $t_a(u0) = t_b(u0) = a$ and $t_a(u1) = t_b(u1) = b$ for any $u \in \{0, 1\}^*$. 

And $\mathcal{A}$ as the safety automaton $(\{q_{in}, q_{b}\}, \{(a, \neq), (b, \neq)\}, q_{in}, \Delta_{\mathcal{A}})$ where $\Delta = \{(q_{in}, (a, \neq), q_{in}, t_b), (q_{in}, (a, \neq), q_{b}, t_b), (q_{b}, (b, \neq), t_a, t_b)\}$. $|L(\widehat{\mathcal{A}})| = \aleph_0$ but $L^{con}(\mathcal{A}) = \{t_a\}$.

Consider $\mathcal{B}$ that (note that $|L(\widehat{\mathcal{B}})| = 2^{\aleph_0}$):

- Checks that the leftmost branch is labelled only by $c$’s.
- Checks that any right subtree of a node on that branch is such that the root is labelled by $c$, the left subtree is $t_a$ while the right subtree is accepted by the automaton $\mathcal{A}$.
Example of Execution

$|L(\hat{A})| = \aleph_0$ but $L^{\text{con}}(A) = \{t_a\}$.

\[ B = (Q_B, \{(a, \neq), (b, \neq), (c, \neq)\}, q_c, \Delta_B, \text{Col}) \text{ with } Q_B = Q_A \cup \{q_c, q'_c\} \]
\[ \text{and } \Delta_B = \Delta_A \cup \{(q_c, (c, \neq), q_c, q'_c), (q'_c, (c, \neq), t_a, q_{\text{in}})\}. \]

Previous Algorithm:

- First detects that $|L(\widehat{B}_{q_{\text{in}}})| \leq \aleph_0$, computes $\kappa(q_{\text{in}}) = 1$ and change $B$ to $B_{q_{\text{in}} \rightarrow t_a}$.
- Then detects that $|L(\widehat{B}_{q'_c})| \leq \aleph_0$, computes $\kappa(q'_c) = 0$ and change $B$ to $B_{q'_c \rightarrow \emptyset}$.
- Finally detects that $\kappa(q_c) = 0$. 
Theorem. The algorithm returns the correct cardinality profile.
Büchi Case

Theorem. The algorithm returns the correct cardinality profile.

Proof ingredients.
- At any stage the language (with contraints) is unchanged.
  - Countable values are correct.
Büchi Case

**Theorem.** The algorithm returns the correct cardinality profile.

**Proof ingredients.**

- Countable values are correct.
- Define run-tree with holes as pieces of runs where:
  - Holes correspond to states where $\kappa$ equals $2^{\aleph_0}$.
  - Parts without holes are accepting and satisfies the constraints.

- Prove that for every state $q$ with $\kappa(q) = 2^{\aleph_0}$ and every $N \geq 0$ there are $N$ $q$-run-tree with holes that are pairwise different.
Theorem. The algorithm returns the correct cardinality profile.

Proof ingredients.

- Countable values are correct.
  - Define run-tree with holes as pieces of runs where:
    - Holes correspond to states where $\kappa$ equals $2^{\aleph_0}$.
    - Parts without holes are accepting and satisfies the constraints.
  - Prove that for every state $q$ with $\kappa(q) = 2^{\aleph_0}$ and every $N \geq 0$ there are $N q$-run-tree with holes that are pairwise different.
  - Combine them to obtain uncountably many accepted trees.
Büchi Case

**Theorem.** The algorithm returns the correct cardinality profile.

**Proof ingredients.**

→ Countable values are correct.

- Define run-tree with holes as pieces of runs where:
  - Holes correspond to states where \( \kappa \) equals \( 2^{\aleph_0} \).
  - Parts without holes are accepting and satisfies the constraints.

- Prove that for every state \( q \) with \( \kappa(q) = 2^{\aleph_0} \) and every \( N \geq 0 \) there are \( N \) \( q \)-run-tree with holes that are **pairwise different**.

- Combine them to obtain uncountably many accepted trees.
Büchi Case

Theorem. The algorithm returns the correct cardinality profile.

Proof ingredients.

- Countable values are correct.
  - Define run-tree with holes as pieces of runs where:
    - Holes correspond to states where \( \kappa \) equals \( 2^{\aleph_0} \).
    - Parts without holes are accepting and satisfies the constraints.
  - Prove that for every state \( q \) with \( \kappa(q) = 2^{\aleph_0} \) and every \( N \geq 0 \) there are \( N \) \( q \)-run-tree with holes that are pairwise different.
  - Combine them to obtain uncountably many accepted trees.
Büchi Case

**Theorem.** The algorithm returns the correct cardinality profile.

**Proof ingredients.**

- Countable values are correct.

- Define run-tree with holes as pieces of runs where:
  - Holes correspond to states where $\kappa$ equals $2^{\aleph_0}$.
  - Parts without holes are accepting and satisfies the constraints.

- Prove that for every state $q$ with $\kappa(q) = 2^{\aleph_0}$ and every $N \geq 0$ there are $N$ $q$-run-tree with holes that are **pairwise different**.

- Combine them to obtain uncountably many accepted trees.

- For Büchi condition do the same but consider only run-tree with holes s.t. a final state occurs in any path from the root to a hole.
**Does it Also Work for co-Büchi?**

No :-( as there exists a co-Büchi automaton $A$ s.t. $|L(A_q)| = 2^\aleph_0$ for all $q$ while $L^{con}(A_q) = \emptyset$. 

- There is at most one possible run per tree: the one that assigns $q_x$ to each node labelled by $(x, \neq)$.
- The unconstrained language from state $q_x$ is the set of all trees such that the root is labelled by $x$ and such that any branch contains finitely many $b$’s: Uncountable.
- But $L^{con}(A_q) = \emptyset$ for $x \in \{a, b\}$. Indeed:
  - An accepted tree would contain at least one node $u_1$ labelled by $b$ (to satisfy $\neq$ at the root).
  - Same for the subtree rooted at $u_1$, and so on…
  - Hence there is $u_1 \triangleright u_2 \triangleright u_3$ all labelled by $b$, leading to violate co-Büchi condition.
Does it Also Work for co-Büchi?

No :-( as there exists a co-Büchi automaton $\mathcal{A}$ s.t. $|L(\mathcal{A}_q)| = 2^{\aleph_0}$ for all $q$ while $L^{\text{con}}(\mathcal{A}_q) = \emptyset$.

Define $\mathcal{A} = (\{q_a, q_b\}, \{a, b\}, q_a, \Delta, \text{Col})$ where $\text{Col}(q_a) = 2$ and $\text{Col}(q_b) = 1$, and $\Delta$ consists of those transitions $(q_x, (x, \neq), q_0, q_1)$ where $x \in \{a, b\}$ and $q_0, q_1$ are any states.

- There is at most one possible run per tree: the one that assigns $q_x$ to each node labelled by $(x, \neq)$.
- The unconstrained language from state $q_x$ is the set of all trees such that the root is labelled by $x$ and such that any branch contains finitely many $b$’s $\Rightarrow$ Uncountable.
Does it Also Work for co-Büchi?

No :-(( as there exists a co-Büchi automaton $\mathcal{A}$ s.t. $|L(\mathcal{A}_q)| = 2^{\aleph_0}$ for all $q$ while $L^{con}(\mathcal{A}_q) = \emptyset$.

Define $\mathcal{A} = (\{q_a, q_b\}, \{a, b\}, q_a, \Delta, \text{Col})$ where $\text{Col}(q_a) = 2$ and $\text{Col}(q_b) = 1$, and $\Delta$ consists of those transitions $(q_x, (x, \neq), q_0, q_1)$ where $x \in \{a, b\}$ and $q_0, q_1$ are any states.

- There is at most one possible run per tree: the one that assigns $q_x$ to each node labelled by $(x, \neq)$.
- The unconstrained language from state $q_x$ is the set of all trees such that the root is labelled by $x$ and such that any branch contains finitely many $b$’s $\Rightarrow$ Uncountable.
- But $L^{con}(\mathcal{A}_{q_x}) = \emptyset$ for $x \in \{a, b\}$. Indeed:
  - An accepted tree would contain at least one node $u_1$ labelled by $b$ (to satisfy $\neq$ at the root).
  - Same for the subtree rooted at $u_1$, and so on...
  - Hence there is $u_1 \sqsubset u_2 \sqsubset u_3 \cdots$ all labelled by $b$, leading to violate co-Büchi condition.
How to Handle the co-Büchi Case? (1/2)

**Trace:** pair $\rho = (t_\rho, r_\rho)$ where $t_\rho$ is an infinite **valid** tree and $r_\rho$ is a run of $A$ on $\overline{t_\rho}$. starting from some arbitrary state. The trace is accepting if the run is.
How to Handle the co-Büchi Case? (1/2)

**Trace:** pair $\rho = (t_\rho, r_\rho)$ where $t_\rho$ is an infinite **valid** tree and $r_\rho$ is a run of $A$ on $t_\rho \approx$. starting from some arbitrary state. The trace is accepting if the run is.

We define two (monotone) operations on sets of traces,

$\text{Attr}(X) = \{(t_\rho, r_\rho) \mid \forall \text{ infinite branch } \pi, \exists u \sqsubseteq \pi \text{ s.t. } (t_\rho[u], r_\rho[u]) \in X\}$

$\text{Safety}(X) = \{(t_\rho, r_\rho) \mid \forall \text{ infinite branch } \pi, \forall u \sqsubseteq \pi, \text{ Col}(r_\rho(u)) = 2, \text{ or } \exists u \sqsubseteq \pi \text{ s.t. } (t_\rho[u], r_\rho[u]) \in X \text{ and Col}(r_\rho(v)) = 2 \forall v \sqsubseteq u\}$

and an increasing transfinite sequence $(X_\alpha)_\alpha$

$$
\begin{align*}
X_0 &= \emptyset \\
X_{\alpha+1} &= \text{Attr}(\text{Safety}(X_\alpha)) \\
X_\alpha &= \bigcup_{\beta < \alpha} X_\beta \quad \text{for } \alpha \text{ limit ordinal}
\end{align*}
$$
How to Handle the co-Büchi Case? (1/2)

**Trace:** pair $\rho = (t_\rho, r_\rho)$ where $t_\rho$ is an infinite valid tree and $r_\rho$ is a run of $A$ on $t_\rho$. starting from some arbitrary state. The trace is accepting if the run is.

We define two (monotone) operations on sets of traces,

$$\textbf{Attr}(X) = \{(t_\rho, r_\rho) \mid \forall \text{ infinite branch } \pi, \exists u \sqsubseteq \pi \text{ s.t. } (t_\rho[u], r_\rho[u]) \in X\}$$

$$\textbf{Safety}(X) = \{(t_\rho, r_\rho) \mid \forall \text{ infinite branch } \pi, \text{ either } \forall u \sqsubseteq \pi, \text{ Col}(r_\rho(u)) = 2, \text{ or } \exists u \sqsubseteq \pi \text{ s.t. } (t_\rho[u], r_\rho[u]) \in X \text{ and } \text{Col}(r_\rho(v)) = 2 \forall v \sqsubseteq u\}$$

and an increasing transfinite sequence $(X_\alpha)_{\alpha}$

$$\begin{cases} 
X_0 = \emptyset \\
X_{\alpha+1} = \text{Attr}(\text{Safety}(X_\alpha)) \\
X_\alpha = \bigcup_{\beta < \alpha} X_\beta 
\end{cases} \text{ for } \alpha \text{ limit ordinal}$$

**Lemma.** The limit of $(X_\alpha)_{\alpha}$ is the set of accepting traces.
How to Handle the co-Büchi Case? (2/2)

Work with the **infinity profile** $p_A$ of $A$

$$p_A(q) = \begin{cases} 
    L^{con}(A_q) & \text{if } |L^{con}(A_q)| < \infty \\
    \infty & \text{otherwise}
\end{cases}$$

**Lemma.** One can compute the cardinality profile from $p_A$. 
How to Handle the co-Büchi Case? (2/2)

Work with the **infinity profile** $p_A$ of $A$

$$p_A(q) = \begin{cases} 
L^{\text{con}}(A_q) & \text{if } |L^{\text{con}}(A_q)| < \infty \\
\infty & \text{otherwise}
\end{cases}$$

**Lemma.** One can compute the cardinality profile from $p_A$.

A **profile** is some $p : Q \to 2^{\text{RegTrees}} \cup \{\infty\}$ that is smaller than $p_A$. 
How to Handle the co-Büchi Case? (2/2)

Work with the **infinity profile** $p_A$ of $A$

$$p_A(q) = \begin{cases} 
L_{con}(A_q) & \text{if } |L_{con}(A_q)| < \infty \\
\infty & \text{otherwise}
\end{cases}$$

**Lemma.** One can compute the cardinality profile from $p_A$.

A **profile** is some $p : Q \to 2^{\text{RegTrees}} \cup \{\infty\}$ that is smaller than $p_A$.

Define a profile counterpart of operators Attr and Safety and show that one can compute $\text{Attr}(p)$ and $\text{Safety}(p)$ from $p$.
How to Handle the co-Büchi Case? (2/2)

Work with the **infinity profile** $p_A$ of $A$

$$p_A(q) = \begin{cases} 
L^\text{con}(A_q) & \text{if } |L^\text{con}(A_q)| < \infty \\
\infty & \text{otherwise}
\end{cases}$$

**Lemma.** One can compute the cardinality profile from $p_A$.

A **profile** is some $p : Q \rightarrow 2^{\text{RegTrees}} \cup \{\infty\}$ that is smaller than $p_A$.

Define a profile counterpart of operators Attr and Safety and show that one can compute Attr$(p)$ and Safety$(p)$ from $p$.

**Lemma.** Let $p$ be a profile. If $p = \text{Attr(Safety}(p))$ then $p = p_A$. 
How to Handle the co-Büchi Case? (2/2)

Work with the **infinity profile** $p_A$ of $A$

$$p_A(q) = \begin{cases} 
L^{con}(A_q) & \text{if } |L^{con}(A_q)| < \infty \\
\infty & \text{otherwise}
\end{cases}$$

**Lemma.** One can compute the cardinality profile from $p_A$.

A **profile** is some $p : Q \rightarrow 2^{RegTrees} \cup \{\infty\}$ that is smaller than $p_A$.

Define a profile counterpart of operators Attr and Safety and show that one can compute $\text{Attr}(p)$ and $\text{Safety}(p)$ from $p$.

**Lemma.** Let $p$ be a profile. If $p = \text{Attr}(\text{Safety}(p))$ then $p = p_A$.

To converge, add a speed-up operator on profiles: $p \mapsto \text{SpeedUp}_p(p)$. 
How to Handle the co-Büchi Case? (2/2)

Work with the **infinity profile** $p_A$ of $A$

$$
p_A(q) = \begin{cases} 
L^{con}(A_q) & \text{if } |L^{con}(A_q)| < \infty \\
\infty & \text{otherwise}
\end{cases}
$$

**Lemma.** One can compute the cardinality profile from $p_A$.

A **profile** is some $p : Q \rightarrow 2^{\text{RegTrees}} \cup \{\infty\}$ that is smaller than $p_A$.

Define a profile counterpart of operators Attr and Safety and show that one can compute Attr($p$) and Safety($p$) from $p$.

**Lemma.** Let $p$ be a profile. If $p = \text{Attr}(\text{Safety}(p))$ then $p = p_A$.

To converge, add a speed-up operator on profiles: $p \mapsto \text{SpeedUp}(p)$.

**Lemma.** Let $p_0$ be the profile that maps $\emptyset$ to every state and let, for any $i \geq 0$, $p_{i+1} = \text{SpeedUp}(\text{Attr}(\text{Safety}(p_i)))$.

Then $(p_i)_{i \geq 0}$ converges in a **finite** number of steps to $p_A$. 
CONCLUSION

Main Contribution: a class of languages of infinite trees that:

- Encompass regular languages.
- Form a Boolean algebra.
- Have interesting expressive power.
- Enjoy a decidable cardinality problem.

Further Work:
- Simplify the proof for the parity case.
- Investigate other decision problems, e.g., the regularity problem.
- Find automata models with decidable emptiness that capture extension of MSO with isomorphism tests.
- Look for applications.
Conclusion

Main Contribution: a class of languages of infinite trees that:

- Encompass regular languages.
- Form a Boolean algebra.
- Have interesting expressive power.
- Enjoy a decidable cardinality problem.

Further Work:

- Simplify the proof for the parity case.
- Investigate other decision problems, e.g. the regularity problem.
- Find automata models with decidable emptiness that capture extension of MSO with isomorphism tests.
- Look for applications.