(Co)-Algebraic and Analytic Aspects of Weighted Automata Minimization and Equivalence, Part I.

Alexandra Silva

University College London
Weighted automata

- Generalization of classical automata

- Interesting applications: speech processing, image recognition, information theory, machine learning, ...

- Fruitful research area since the 60’s

Recent (co)algebraic research

- **Minimization** [Boreale, Bonchi, Bonsangue & Silva 2011; Adamek, Bonchi, Huelbusch, Koenig, Milius, Silva 2012; Koenig & Kupper 2014; …]

- **Equivalence — proof methods** [Silva 2010, Milius 2010, Bonsangue, Milius, Silva 2013; Urabe & Hasuo 2014a,b; …]

- **Decidability** [Bonsangue, Milius, Silva 2013; Milius 2015; …]
This tutorial

- Weighted automata as coalgebras
- Weighted bisimulation
- Language equivalence
Weighted automata intuitively

Intuition: takes a word, runs it through the automaton and computes a weight
Weighted automata intuitively

Intuition: takes a word, runs it through the automaton and computes a weight
Weighted automata intuitively

Intuition: takes a word, runs it through the automaton and computes a weight
Weighted automata intuitively

Intuition: takes a word, runs it through the automaton and computes a weight

\[
\begin{align*}
\text{a} &\mapsto \frac{3}{2} \times 2 - \frac{3}{2} \times 2 + \frac{1}{2} \times 2 = 1
\end{align*}
\]
Weighted automata intuitively

\[ a \mapsto \frac{3}{2} \times 2 - \frac{3}{2} \times 2 + \frac{1}{2} \times 2 = 1 \]

Intuition: takes a word, runs it through the automaton and computes a weight
Weighted automata intuitively

Intuition: takes a word, runs it through the automaton and computes a weight

\[
\begin{align*}
a & \mapsto \frac{3}{2} \times 2 - \frac{3}{2} \times 2 + \frac{1}{2} \times 2 = 1 \\

aa & \mapsto \frac{3}{2} \left( \frac{3}{2} \times 2 - \frac{3}{2} \times 2 + \frac{1}{2} \times 2 \right) \\
& - \frac{3}{2} \times \left( -\frac{1}{2} \times 2 + \frac{1}{2} \times 2 + \frac{1}{2} \times 2 \right) \\
& + \frac{1}{2} \times (1 \times 2) = 1
\end{align*}
\]
Weighted automata intuitively

A:4 Bonsangue, Milius, Silva  

Syntactic expressions modulo the axioms and rules of the calculus form the rational fixpoint \( \bar{F} \) for the lifting of the functor \( F \) to \( T \)-algebras.  

Then we apply our abstract results to the monad \( V \) of free semimodules for a Noetherian semiring \( S \) and the functor \( FX = S \times X \), where \( A \) is a finite input alphabet, and we show how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section ??, and, as a special case, of non-deterministic automata in Section ??.

Weighted automata were introduced by Schützenberger [?], see also [?]. For example, take the following two weighted automata over the alphabet \( A = \{ a \mapsto b \mapsto c \mapsto d \} \) with weight over the semiring of natural numbers (output values in the states are represented with a double arrow, when omitted they are zero):

- \( a, \frac{1}{2} \mapsto 2 \)
- \( a, \frac{1}{2} \mapsto a, 2 \mapsto 2 \)
- \( a, \frac{1}{2} \mapsto a, -\frac{1}{2} \mapsto 2 \)
- \( a, \frac{1}{2} \mapsto a, -\frac{1}{2} \mapsto 2 \)
- \( a, \frac{1}{2} \mapsto a, -\frac{1}{2} \mapsto 2 \)
- \( a, \frac{1}{2} \mapsto a, -\frac{1}{2} \mapsto 2 \)
- \( a, 1 \mapsto 2 \)
- \( a, 1 \mapsto 2 \)

We will see in Section ?? that they are coalgebras for the composition of the functor \( FX = S \times X \) with the monad \( V \) of free semimodules for the semiring \( S \) of natural numbers. What is interesting is that the leftmost states of these automata are not bisimilar, but they recognise the same weighted language. Namely, the language that associates with each word \( a^n (bc)^n \) the weight \( 2 \cdot 6^n \), with \( a^n (bc)^n d \) the weight \( 2 \cdot 6^n \cdot 4 \) and with any other word weight zero. We will provide an algebraic proof of this equivalence in the sequel. To give upfront the reader a feeling for how intricate it can get to reason about weighted language equivalence, we show another example of two weighted automata over the singleton alphabet \( A = \{ a \} \) but with weight over the field of real numbers.

Conjecture: \( \bullet \) assigns 1 to every word \( a^n \)
Weighted automata intuitively

Conjecture: \( \bullet \) assigns 1 to every word \( a^n \)
Weighted automata - formally

Weights: semiring \((\mathbb{S}, +, \cdot, 0, 1)\)

- Reals with usual addition and multiplication
- Natural Numbers with usual addition and multiplication
- Tropical semiring \((\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)\)
A weighted automaton over alphabet $A$ is a tuple $(S, \alpha, (T)_{a \in A}, \beta)$

- (finite) set of states
- input vector — initial state
- transition matrices
- output vector — final state
More detailed, the syntactic expressions of our calculus are defined by the grammar. An algebraic proof is rather simple and instructive. For the right-hand automaton one needs some more ingenuity. We shall see that the automaton, it is still relatively easy to convince oneself that this is the case, whereas to the empty word weight of real numbers.

The leftmost states of these automata recognize the weighted language that assigns to the empty word weight 0. We will provide an algebraic proof of this fact. 

What is interesting is that the leftmost states of these automata are not bisimilar, but they recognize the same weighted language. Namely, the language that associates with each word $a^n$ the weight $n$.

We start with the calculus for weighted bisimilarity obtained from the generic ex-

We illustrate our calculus with the following two weighted automata over the alphabet $\{a\}$, and with any other word weight zero. We will provide an algebraic proof of this fact.

Then we apply our abstract results to the monad of free semimodules for the semiring of natural numbers. What is interesting is that the leftmost states of these automata are not bisimilar, but they recognise the same weighted language. Namely, the language that they are coalgebras for the composition of the functor $\mathcal{A}$.

In the left-hand automaton, it is still relatively easy to convince oneself that this is the case, whereas to the empty word weight of real numbers.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.

We will see in Section 6 how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section 5.
Language semantics

\[
\llbracket - \rrbracket : S \rightarrow (A^* \rightarrow S)
\]

\[
\llbracket s \rrbracket (a_1 \cdots a_k) = \beta \times T_{a_1} \times \cdots \times T_{a_k} \times \alpha^T
\]
Language semantics

\[ [-] : S \rightarrow (A^* \rightarrow S) \]

\[ [s](a_1 \cdots a_k) = \beta \times T_{a_1} \times \cdots \times T_{a_k} \times \alpha^T \]

\[
\begin{bmatrix}
2 & 2 & 2 \\
\end{bmatrix}
\begin{pmatrix}
3/2 & 1/2 & 0 \\
-3/2 & -1/2 & 0 \\
1/2 & 1/2 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
\end{pmatrix}
= 1
\]
Language semantics

\[
\begin{pmatrix}
\frac{3}{2} & \frac{1}{2} & 0 \\
-\frac{3}{2} & -\frac{1}{2} & 0 \\
\frac{1}{2} & \frac{1}{2} & 1
\end{pmatrix}^n = \begin{pmatrix}
\frac{3}{2} & \frac{1}{2} & 0 \\
-\frac{3}{2} & -\frac{1}{2} & 0 \\
\frac{1}{2} & \frac{1}{2} & 1
\end{pmatrix}
\]
Language semantics

\[
\begin{pmatrix}
\frac{3}{2} & \frac{1}{2} & 0 \\
-\frac{3}{2} & -\frac{1}{2} & 0 \\
\frac{1}{2} & \frac{1}{2} & 1
\end{pmatrix}^n = \begin{pmatrix}
\frac{3}{2} & \frac{1}{2} & 0 \\
-\frac{3}{2} & -\frac{1}{2} & 0 \\
\frac{1}{2} & \frac{1}{2} & 1
\end{pmatrix}
\]

\[
\begin{bmatrix}
2 & 2 & 2
\end{bmatrix} \times \begin{pmatrix}
\frac{3}{2} & \frac{1}{2} & 0 \\
-\frac{3}{2} & -\frac{1}{2} & 0 \\
\frac{1}{2} & \frac{1}{2} & 1
\end{pmatrix}^n \times \begin{pmatrix}
1 \\
0 \\
0
\end{pmatrix} = 1
\]
Another type of semantics

An equivalence relation \( R \subseteq X \times X \) is a weighted bisimulation if for all \((x_1, x_2) \in R\), it holds that:

1. \( \beta(x_1) = \beta(x_2) \),

2. \( \forall a \in A, \ x' \in X, \sum_{x'' \in [x']_R} T_a(x_1, x'') = \sum_{x'' \in [x']_R} T_a(x_2, x'') \).

Bisimilarity — example
Bisimilarity vs language
Weighted automata, coalgebraically I

\[ S^X = \{ \varphi \mid \varphi: X \to S \text{ with finite support} \} \]

\[
h: X \to Y
\]

\[
S^h(\varphi)(y) = \sum_{h(x) = y} \varphi(x)
\]
Weighted automata, coalgebraically I

A weighted automaton is a coalgebra for the functor

\[ W(X) = S \times (S^X)^A \]
A weighted automaton is a coalgebra for the functor

\[ \mathcal{W}(X) = S \times (S^X)^A \]

\[ X \to S \times (S^X)^A \]

\[ \beta: X \to S \]

\[ T_a: X \to S^X, \quad \forall a \in A \]
A weighted automaton is a coalgebra for the functor

\[ \mathcal{W}(X) = S \times (S^X)^A \]

Coalgebraic modelling does not include the initial vector

\[ X \rightarrow S \times (S^X)^A \]

\[ T_a : X \rightarrow S^X, \quad \forall a \in A \]
Some facts about $\mathcal{W}$

$\mathcal{W}$ is a bounded functor (Gumm’01)

There exists a final $\mathcal{W}$ coalgebra

$\mathcal{W}$ does not preserve weak pullbacks
Behavioral equivalence

\[ \begin{align*}
X & \xrightarrow{!} \Omega \\
\mathcal{W}(X) & \xrightarrow{\mathcal{W}(!)} \mathcal{W}(\Omega) \\
\langle \beta, T \rangle & \xrightarrow{\mathcal{W}} \mathcal{W}(\Omega)
\end{align*} \]

\( x \approx y \iff !(x) = !(y) \)

**Theorem**

\( x \) is (Bucholz) weighted bisimilar to \( y \) iff \( x \approx y \)
How about Aczel-Mendler bisimilarity?

\[
\begin{align*}
X & \xleftarrow{\beta_X} T_X \\
\mathcal{W}(X) & \xleftarrow{\mathcal{W}(\pi_1)} \mathcal{W}(R) \\
\mathcal{W}(X) & \xleftarrow{\mathcal{W}(\beta_X)} \mathcal{W}(\pi_1) \\
R & \xrightarrow{\beta_R} T_R \\
\mathcal{W}(R) & \xrightarrow{\mathcal{W}(\pi_2)} \mathcal{W}(Y) \\
\mathcal{W}(R) & \xrightarrow{\mathcal{W}(\beta_R)} \mathcal{W}(\pi_2) \\
Y & \xrightarrow{o_Y, t_Y} Y
\end{align*}
\]

**Notation:** $x$ and $y$ are AM-bisimilar iff there exists $R$ as above;

\[x \sim y\]
Bisimilarity vs Beh. equiv

\[ x_1 \cong x_2 \text{ but } x_1 \not\sim y_1 \]
Bisimilarity vs Beh. equiv

- Transitions with negative numbers play a key role
- Similar examples can be constructed for commutative monoids that are not zero-sum free (Gumm’ 01)
Weighted automata, coalgebraically II

For simplicity let us take $S = \mathbb{F}$ (field).

$\textbf{Vect}$ — finite dimensional vector spaces and linear maps

**Linear weighted automata**: coalgebra in $\textbf{Vect}$ for the functor

$$\mathcal{L}(V) = \mathbb{F} \times V^A$$
Semantics of LWA

\[ \gamma : \mathbb{F}^A^* \rightarrow \mathbb{F} \quad \gamma(f) = f(\varepsilon) \]

\[ T_a : \mathbb{F}^A^* \rightarrow \mathbb{F}^A^* \quad T_a(f)(w) = f(a \cdot w) \]

\[ < \gamma, T > : \mathbb{F}^A^* \rightarrow \mathcal{L}(\mathbb{F}^A^*) \]
Semantics of LWA

\[ <\gamma, T>: F^{A^*} \rightarrow \mathcal{L}(F^{A^*}) \]

is the final coalgebra in \( \textbf{Vect} \)

\[
\begin{array}{ccc}
V & \xrightarrow{1} & F^{A^*} \\
\downarrow & & \downarrow <\gamma, T> \\
\mathcal{L}(V) & \xrightarrow{<\beta, T>} & \mathcal{L}(F^{A^*})
\end{array}
\]
Semantics of LWA

\[ \langle \gamma, T \rangle : \mathbb{F}^{A^*} \to \mathcal{L}(\mathbb{F}^{A^*}) \] is the final coalgebra in \( \textbf{Vect} \)

\[
\begin{align*}
V & \xrightarrow{1} \mathbb{F}^{A^*} \\
\langle \beta,T \rangle \downarrow & \quad \downarrow \langle \gamma,T \rangle \\
\mathcal{L}(V) & \xrightarrow{\quad} \mathcal{L}(\mathbb{F}^{A^*})
\end{align*}
\]

From the commutativity one can compute for \( \alpha \in V \)

\[ l(\alpha)(a_1 \cdots a_k) = \beta \times T_{a_1} \times \cdots \times T_{a_k} \times \alpha^T \]
Semantics of LWA

\[ \langle \gamma, T \rangle : F^{A^*} \rightarrow L(F^{A^*}) \]

is the final coalgebra in \( \textbf{Vect} \)

From the commutativity one can compute for \( \alpha \in V \)

\[ l(\alpha)(a_1 \cdots a_k) = \beta \times T_{a_1} \times \cdots \times T_{a_k} \times \alpha^T \]

Language semantics
### Mismatch

<table>
<thead>
<tr>
<th>Weighted automata</th>
<th>Linear weighted automata</th>
</tr>
</thead>
<tbody>
<tr>
<td>weighted bisimulation</td>
<td>language semantics</td>
</tr>
<tr>
<td>states</td>
<td>vectors</td>
</tr>
</tbody>
</table>

\[ V \rightarrow S \times V^A \]
## Mismatch

<table>
<thead>
<tr>
<th>Weighted automata</th>
<th>Linear weighted automata</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \rightarrow S \times (S^X)^A$</td>
<td>$V \rightarrow S \times V^A$</td>
</tr>
<tr>
<td>weighted bisimulation</td>
<td>language semantics</td>
</tr>
<tr>
<td>states</td>
<td>vectors</td>
</tr>
</tbody>
</table>
Mismatch

Weighted automata

Linear weighted automata

$V \rightarrow S \times V^A$

weighted bisimulation

language semantics

states

vectors
## Mismatch

<table>
<thead>
<tr>
<th>Weighted automata</th>
<th>Linear weighted automata</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \rightarrow S \times (S^X)^A$</td>
<td>$V \rightarrow S \times V^A$</td>
</tr>
</tbody>
</table>

- Weighted bisimulation
- Language semantics
- States
- Vectors
Combining both worlds

\[
\begin{align*}
X & \downarrow \langle \beta, T \rangle \\
F \times (F^X)^A &
\end{align*}
\]
Combining both worlds

\[ \begin{array}{c}
X \xrightarrow{e} FX \\
\langle \beta, T \rangle \\
\downarrow \\
F \times (F^X)^A
\end{array} \]
Combining both worlds

\[ X \rightarrow^e \mathbb{F} X \]

\[ (\mathbb{F} X)^A \]

\[ \mathbb{F} X \text{ free vector space generated by } X \]

\[ \begin{pmatrix} x_1 & (r_1) \\ \vdots & \vdots \\ x_n & (r_n) \end{pmatrix} \]
Combining both worlds

\[ X \xrightarrow{e} \mathbb{F}X \]

\[ \mathbb{F} \times (\mathbb{F}X)^A \]

\( \langle \beta, T \rangle \)

\( \langle \hat{\beta}, \hat{T} \rangle \)

\( \mathbb{F}X \) free vector space generated by \( X \)

\[
\begin{pmatrix}
x_1 & \left( r_1 \right) \\
\vdots & \vdots \\
x_n & \left( r_n \right)
\end{pmatrix}
\]
Combining both worlds

\[ \mathbb{F}^X \text{ free vector space generated by } X \]

\[ x_1 \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} \]

\[ <\hat{\beta}, \hat{T}> \quad \text{is the unique linear extension of} \quad <\beta, T> \]
Combining both worlds

\[ \mathbb{F}^X \text{ free vector space generated by } X \]

\[ X \xrightarrow{e} \mathbb{F}^X \]

\[ \langle \beta, T \rangle \]

\[ \mathbb{F} \times (\mathbb{F}^X)^A \]

\[ \langle \hat{\beta}, \hat{T} \rangle \]

\[ \langle \hat{\beta}, \hat{T} \rangle \text{ is the unique linear extension of } \langle \beta, T \rangle \]
Combining both worlds

\[ \mathbb{F}^X \text{ free vector space generated by } X \]

\[ X \xrightarrow{e} \mathbb{F}^X \]

\[ <\beta,T> \downarrow \quad \downarrow \quad <\hat{\beta},\hat{T}> \]

\[ \mathbb{F} \times (\mathbb{F}^X)^A \]

\[ \hat{\beta} \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} = \beta \times \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}^T \]

\[ <\hat{\beta},\hat{T}> \text{ is the unique linear extension of } <\beta,T> \]
Combining both worlds

\[ \mathbb{F}^X \text{ free vector space generated by } X \]

\[ \xymatrix{ X \ar[d]_{<\beta,T>} \ar[r]^e & \mathbb{F}X \ar[d]_{<\hat{\beta},\hat{T}>} \\
\mathbb{F} \times (\mathbb{F}^X) & & \hat{T}_a \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} = T_a \times \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}^T } \]

\[ <\hat{\beta},\hat{T}> \text{ is the unique linear extension of } <\beta,T> \]
Semantics

\[
\begin{align*}
X & \xrightarrow{e} FX & \cdots & \xrightarrow{1} FA^* \\
<\beta, T> & \downarrow & \cdots & <\gamma, T> \\
F \times (FX)_A & \xleftarrow{<\hat{\beta}, \hat{T}>} & \cdots & \xrightarrow{\mathcal{L}(FA^*)}
\end{align*}
\]
Semantics

\[ X \xrightarrow{e} FX \xrightarrow{l} FA^* \]

\[ F \times (FX)^A \xrightarrow{\text{ker}(I)} \mathcal{L}(FA^*) \]

\[ \ker(I) \text{ is exactly weighted language equivalence} \]

\[ x_1 \approx x_2 \iff l(e(x_1)) = l(e(x_2)) \]
Computing the kernel of $I$

**Three methods** [Boreale, Bonchi, Bonsangue, Rutten & Silva’ I&C 11]

- linear partition refinement — forward
- linear partition refinement — backward
- system of behavioural differential equations (rational streams)
Computing the kernel of $l$

$x_1 \approx x_2 \iff l(e(x_1)) = l(e(x_2))$

**Three methods** [Boreale, Bonchi, Bonsangue, Rutten & Silva’ I&C 11]

- linear partition refinement — forward
- linear partition refinement — backward
- system of behavioural differential equations (rational streams)
Consider the sequence \( (R_i)_{i \geq 0} \) of subspaces of \( V \) defined inductively by

\[
R_0 = \ker(\beta) \quad R_{i+1} = R_i \cap \bigcap_{a \in A} T_a(R_i)^{-1}
\]

\( T_a(R_i)^{-1} \) is the space \( \{ v \in V \mid T_a(v) \in R_i \} \).
Consider the sequence $(R_i)_{i \geq 0}$ of subspaces of $V$ defined inductively by

$$R_0 = \ker(\beta) \quad R_{i+1} = R_i \bigcap \bigcap_{a \in A} T_a(R_i)^{-1}$$

$T_a(R_i)^{-1}$ is the space $\{v \in V \mid T_a(v) \in R_i\}$.

There is $j \leq \dim(V)$ such that $R_{j+1} = R_j$. The largest linear weighted bisimulation is $\approx_L = R_j$. 

Linear partition refinement — forward
Linear partition refinement example

\[ \beta = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix} \]

\[ T_a = \begin{pmatrix} 1 & \frac{1}{3} & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \]

\[ T_b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} \]
Linear partition refinement example

\[ B_0 = \left\{ \left( \begin{array}{c} -\frac{1}{2} \\ 1 \\ 0 \end{array} \right), \left( \begin{array}{c} -\frac{1}{2} \\ 0 \\ 1 \end{array} \right) \right\}. \quad \text{Basis for } \ker(\beta) \]

\[ \beta = \left( \begin{array}{ccc} 2 & 1 & 1 \end{array} \right) T_a = \left( \begin{array}{ccc} 1 & \frac{1}{3} & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \right) T_b = \left( \begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & \frac{1}{3} & 0 \end{array} \right) \]
Linear partition refinement example

\[ \beta = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix} \]

\[ T_a = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad T_b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} \]

Example 1. Because \( i + B B x = i \{3 \} \) is in general a large subspace: since \( \text{dim} R \in \mathbb{N} \).

\[ \text{We weight automata} \]

Then, \( R_1 \cap R_0 = \{0\} \) (\( R_0 \) in Fig. 5.)

Concretely, the algorithm iteratively computes a basis \( \text{ker}(\beta) \) in the actual computation of the basis of \( \text{ker}(\beta) \). This might be problematic in the actual computation of the basis of \( \text{ker}(\beta) \).

Now, in order to check if \( v \in V \) belongs to \( \text{ker}(\beta) \), we have to look if \( v \in V \) is included in \( \text{ker}(\beta) \). Since the back algorithm represents in the next section which will avoid this problem.

\[ B_0 = \left\{ \begin{pmatrix} -\frac{1}{2} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -\frac{1}{2} \\ 0 \\ 1 \end{pmatrix} \right\}. \] Basis for \( \text{ker}(\beta) \)

\[ B_0^a = \left\{ \begin{pmatrix} -\frac{1}{3} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \right\} \quad \text{and} \quad B_0^b = \left\{ \begin{pmatrix} -\frac{1}{6} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -\frac{3}{2} \\ 0 \\ 1 \end{pmatrix} \right\} . \]
Linear partition refinement example

\[
\beta = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix}
\]

\[T_a = \begin{pmatrix}
1 & \frac{1}{3} & 1 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{pmatrix}
\]

\[T_b = \begin{pmatrix}
1 & 0 & 0 \\
0 & 0 & 3 \\
0 & \frac{1}{3} & 0
\end{pmatrix}
\]

Concretely, the algorithm iteratively computes a basis \(B\) for \(V\) and \(R\) such that \(\dim(Ker(B)) = 2\). Since the back algorithm presented in the next section is

\[B_0 = \left\{ \begin{pmatrix} -\frac{1}{2} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -\frac{1}{2} \\ 0 \\ 1 \end{pmatrix} \right\}. \quad \text{Basis for } \ker(\beta)
\]

\[B_1^a = \left\{ \begin{pmatrix} -\frac{1}{3} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \right\} \quad \text{and} \quad B_1^b = \left\{ \begin{pmatrix} -\frac{1}{6} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -\frac{3}{2} \\ 0 \\ 1 \end{pmatrix} \right\}.
\]

\[B_1 = \left\{ \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} \right\}. \quad R_0 \cap t(R_0)(a)^{-1} \cap t(R_0)(b)^{-1}
\]
Linear partition refinement example

\[ \begin{align*}
\beta &= \begin{pmatrix} 2 & 1 & 1 \end{pmatrix} T_a = \begin{pmatrix} 1 & 1/3 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} T_b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 1/3 & 0 \end{pmatrix}
\end{align*} \]

\[ B_1 = \left\{ \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} \right\} \]

\[ R_0 \cap t(R_0)(a)^{-1} \cap t(R_0)(b)^{-1} \]
Linear partition refinement example

\[ \beta = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix} \]

\[ T_a = \begin{pmatrix} 1 & \frac{1}{3} & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad T_b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} \]

\[ B_2^a = \left\{ \begin{pmatrix} -\frac{1}{3} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\} \quad \text{and} \quad B_2^b = \left\{ \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} \right\}. \]

\[ B_1 = \left\{ \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} \right\}. \quad R_0 \cap t(R_0)(a)^{-1} \cap t(R_0)(b)^{-1} \]
Linear partition refinement example

\[ B_2^a = \left\{ \begin{pmatrix} -\frac{1}{3} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \right\} \quad \text{and} \quad B_2^b = \left\{ \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} \right\}. \]

\[ B_2 = \left\{ \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} \right\} \]

\[ R_1 \cap t(R_1)(a)^{-1} \cap t(R_0)(b)^{-1} \]

\[ R_0 \cap t(R_0)(a)^{-1} \cap t(R_0)(b)^{-1} \]

\[ \beta = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix} \]

\[ T_a = \begin{pmatrix} 1 & \frac{1}{3} & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad T_b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} \]
Linear partition refinement example

\[ B_1 = \left\{ \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} \right\}. \]

To check \( v \approx w \) we need to check \( v - w \in R_1 \)

\[
\beta = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix} \quad T_a = \begin{pmatrix} 1 & \frac{1}{3} & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad T_b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & \frac{1}{3} & 0 \end{pmatrix}
\]
Linear partition refinement example

\[
\begin{align*}
B_1 &= \left\{ \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix} \right\}.
\end{align*}
\]

To check \( v \approx w \) we need to check \( v - w \in R_1 \).

E.g. \( x_1 \approx \frac{3}{2} x_1 + \frac{1}{2} x_3 \)

\[
\beta = (2 \ 1 \ 1) \quad T_a = \begin{pmatrix} 1 & \frac{1}{3} & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad T_b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & \frac{1}{3} & 0 \end{pmatrix}
\]
Some remarks

• There is a tight connection between the forward algorithm and the final sequence of the $\mathcal{L}$ functor

• The backward algorithm avoid the fact that ker( ) is a very large space (and hence computing a base might be tricky)

• The third algorithm in the paper is closely related to formal power series (which were a classical way to study weighted languages)

• What about semirings?
Some remarks

• There is a tight connection between the forward algorithm and the final sequence of the $\mathcal{L}$ functor.

• The backward algorithm avoids the fact that $\text{ker}(\_)$ is a very large space (and hence computing a base might be tricky).

• The third algorithm in the paper is closely related to formal power series (which were a classical way to study weighted languages).

• What about semirings?
Equivalence of weighted automata, algebraically

A: Bonsangue, Milius, Silva

Syntactic expressions modulo the axioms and rules of the calculus form the rational fixpoint $\bar{F}$ for the lifting of the functor $F$ to $T$-algebras.

Then we apply our abstract results to the monad $V$ of free semimodules for a Noetherian semiring $S$ and the functor $FX = S \times X A$, where $A$ is a finite input alphabet, and we show how to obtain a sound and complete calculus for the language equivalence of weighted automata in Section ??, and, as a special case, of non-deterministic automata in Section ??.

Weighted automata were introduced by Schmidt, see also [?]. For example, take the following two weighted automata over the alphabet $A = \{a \downarrow b \downarrow c \downarrow d\}$ with weight over the semiring of natural numbers (output values in the states are represented with a double arrow, when omitted they are zero):

- $c, 6 \downarrow \downarrow$
- $a, 2 / /$
- $\downarrow \downarrow$
- $b, 1 /
- $d, 2 / /$
- $\downarrow \downarrow$
- $12$

We will see in Section ?? that they are coalgebras for the composition of the functor $FX = S \times X A$ with the the monad $V$ of free semimodules for the semiring $S$ of natural numbers. What is interesting is that the leftmost states of these automata are not bisimilar, but they recognise the same weighted language. Namely, the language that associates with each word $a (bc)^n$ the weight $2 \cdot 6^n$, with $a (bc)^n d$ the weight $2 \cdot 6^n \cdot 4$ and with any other word weight zero. We will provide an algebraic proof of this equivalence in the sequel. To give upfront the reader a feeling for how intricate it can get to reason about weighted language equivalence, we show another example of two weighted automata over the singleton alphabet $A = \{a\}$ but with weight over the field of real numbers.
More detailed, the syntactic expressions of our calculus are defined by the grammar expression calculus of [algebraic proof is rather simple and instructive.]

To get to reason about weighted language equivalence, we show another example of two equivalence in the sequel. To give upfront the reader a feeling for how intricate it can be.

Weighted automata were introduced by Schützenberger [and, as a special case, of non-deterministic automata weighted automata in Section A:4 Bonsangue, Milius, Silva].

The leftmost states of these automata recognize the weighted language that assigns to the empty word weight $0$ and to any word $w$ its length $|w|$. What is interesting is that the leftmost states of these automata are natural numbers. What is interesting is that the leftmost states of these automata are natural numbers. What is interesting is that the leftmost states of these automata are natural numbers.
Weighted expressions

\[ E ::= x \mid 0 \mid E \oplus E \mid r \mid a.(r \bullet E) \mid \mu x.E \]
Weighted expressions

\[ E ::= x \mid 0 \mid E \oplus E \mid r \mid a.(r \bullet E) \mid \mu x. E \]

\begin{align*}
E_1 \equiv E_2[E_1/x] & \implies E_1 \equiv \mu x. E_2 \\
(E_1 \oplus E_2) \oplus E_3 & \equiv E \oplus (E_2 \oplus E_3) \\
a.(r \bullet E) \oplus a.(s \bullet E) & \equiv a.((r + s) \bullet E) \\
\end{align*}

\begin{align*}
a.(0 \bullet E) & \equiv 0 \\
0 \oplus E & \equiv 0 \\
E_1 \oplus E_2 & \equiv E_2 \oplus E_1 \\
r \oplus s & \equiv r + s \\
\mu x. E & \equiv E[\mu x. E/x] \\
0 & \equiv 0
\end{align*}
Weighted expressions

\[ E ::= x \mid 0 \mid E \oplus E \mid r \mid a.(r \cdot E) \mid \mu x. E \]

\[
\begin{align*}
E_1 \equiv E_2[E_1/x] & \implies E_1 \equiv \mu x. E_2 \\
(E_1 \oplus E_2) \oplus E_3 & \equiv E \oplus (E_2 \oplus E_3) \\
a.(r \cdot E) \oplus a.(s \cdot E) & \equiv a.((r + s) \cdot E)
\end{align*}
\]

\[
\begin{align*}
a.(0 \cdot E) & \equiv 0 \\
0 \oplus E & \equiv 0 \\
E_1 \oplus E_2 & \equiv E_2 \oplus E_1 \\
r \oplus s & \equiv r + s \\
\mu x. E & \equiv E[\mu x. E/x] \\
0 & \equiv 0
\end{align*}
\]

\[
\begin{align*}
a.(r \cdot (E_1 \oplus E_2)) & \equiv a.(r \cdot E_1) \oplus a.(r \cdot E_2) \\
a.(r \cdot b.(s \cdot E)) & \equiv a.((rs) \cdot b.(1 \cdot E)) \\
a.(r \cdot s) & \equiv a.(1 \cdot rs) \\
a.(r \cdot 0) & \equiv 0.
\end{align*}
\]
Weighted expressions

\[
E ::= x \mid 0 \mid E \oplus E \mid r \mid a.(r \cdot E) \mid \mu x.E
\]

\[
\begin{align*}
E_1 \equiv E_2[E_1/x] & \implies E_1 \equiv \mu x.E_2 \\
(E_1 \oplus E_2) \oplus E_3 & \equiv E \oplus (E_2 \oplus E_3) \\
a.(r \cdot E) \oplus a.(s \cdot E) & \equiv a.((r + s) \cdot E)
\end{align*}
\]

\[
\begin{align*}
a.(0 \cdot E) & \equiv 0 \\
0 \oplus E & \equiv 0 \\
E_1 \oplus E_2 & \equiv E_2 \oplus E_1 \\
r \oplus s & \equiv r + s \\
0 & \equiv 0
\end{align*}
\]

\[
\begin{align*}
a.(r \cdot (E_1 \oplus E_2)) & \equiv a.(r \cdot E_1) \oplus a.(r \cdot E_2) \\
a.(r \cdot b.(s \cdot E)) & \equiv a.((rs) \cdot b.(1 \cdot E))
\end{align*}
\]

\[
\begin{align*}
a.(r \cdot s) & \equiv a.(1 \cdot rs) \\
a.(r \cdot 0) & \equiv 0.
\end{align*}
\]

Sound and complete wrt weighted bisimilarity for noetherian semirings

[Bonsangue, Milius, Silva TOCL 2013]
Back to the example

\[ E_1 = a.(1 \cdot E'') \oplus a.(-1 \cdot E) \oplus 2 \]

\[ E_2 = \mu x.a.(\frac{3}{2} \cdot x) \oplus a.(-\frac{3}{2} \cdot E') \oplus a.(\frac{1}{2} \cdot E'') \oplus 2 \]

\[ E' = \mu z.a.(-\frac{1}{2} \cdot z) \oplus a.(\frac{1}{2} \cdot x) \oplus a.(\frac{1}{2} \cdot E'') \oplus 2 \]

\[ E = \mu y.a.(1 \cdot y) \oplus 1 \]

\[ E'' = \mu y.a.(1 \cdot y) \oplus 2 \]
Back to the example

Using the axioms we can prove

\[ E_1 \equiv E_2 \]

We can also (even more easily) prove

\[ E_1 \equiv \mu x. a.(1 \cdot x) + 1 \]

semantically

\[ a^n \rightarrow 1 \]

\[ E_1 = a. \frac{3}{2} \]

\[ E_2 = \mu x. a.(1 \cdot x) + a.(-\frac{1}{2} \cdot 2) \]

\[ E' = \mu z. a.(-\frac{1}{2} \cdot z) \oplus a.(\frac{1}{2} \cdot x) \oplus a.(\frac{1}{2} \cdot E'') \oplus 2 \]
Some results, some problems

• Algorithms
• (Co)algebraic understanding of decidability
• Probabilistic automata
Conclusions

- Weighted automata are a very powerful generalisation of classical automata

- Algorithms for equivalence (and minimisation) vary depending on the semiring

- (Co)algebra — good abstract framework to study semantics, algorithms, and decidability
Conclusions

• Weighted automata are a very powerful generalisation of classical automata

• Algorithms for equivalence (and minimisation) vary depending on the semiring

• (Co)algebra — good abstract framework to study semantics, **algorithms, and decidability**

Thanks! Questions?