Functional Weighted Automata

Emmanuel Filiot (ULB), Raffaella Gentilini (UPerugia),
Jean-François Raskin (ULB)

Functional Weighted Automata (over finite strings)

Definition

A weighted automaton is **functional** if **all accepting runs** over the same input string have the **same value**.

For functional WA over an idempotent semiring, $\oplus$ is useless.

Semantics

If A is a functional WA and $w \in \Sigma^*$.

$$A(w) = \begin{cases} Value(r) & \text{for } \textbf{some} \text{ accepting run } r \text{ if it exists} \\ \bot & \text{otherwise} \end{cases}$$

Weight Sets and Value Functions (in this talk)

Sum over $\mathbb{Z}$

$V(r) = \sum_{\ell=1}^{n} i_\ell$

Discounted Sum over $\mathbb{Z}$ and $0 < \lambda < 1$ a rational

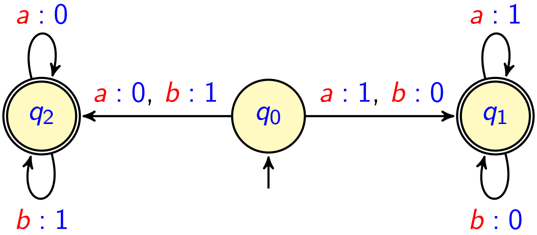
$V(r) = \sum_{\ell=1}^{n} \lambda^{\ell} i_\ell$

Ratio over $\mathbb{N}^2$

$V(r) = \dfrac{\sum_{\ell=1}^{n} r_\ell}{1 + \sum_{\ell=1}^{n} c_\ell}$

Functional vs Non-Functional

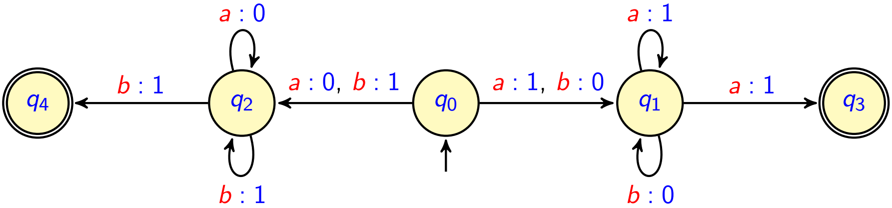
Over the semiring $(\mathbb{N}, max, +)$ and $\Sigma = \{a, b\}$.



$$A(w) = max\ (\#_a(w), \#_b(w))$$

Functional < Non-Functional
(for the 3 measures)

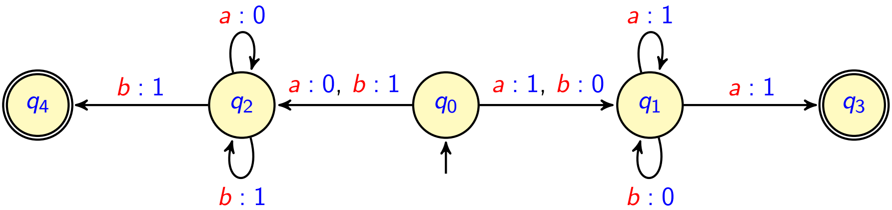
Sequential vs Functional



$$A(w\sigma) = \#_\sigma(w) + 1 \qquad \sigma \in \Sigma$$

Sequential $<$ Functional
(for the 3 measures)

Sequential vs Functional



$$A(w\sigma) = \#_\sigma(w) + 1 \qquad \sigma \in \Sigma$$

Sequential $<$ Functional
(for the 3 measures)

Functional WA are closed under regular look-ahead

Decision Problems

$\perp < \nu$ for all values ν, $\triangleleft \in \{<, \leq\}$

- **functionality:** is A functional ?
- $(\triangleleft\nu)$**-emptiness:** $\exists w \in \Sigma^*, \perp < A(w) \triangleleft \nu$?
- $(\triangleleft\nu)$**-universality:** $\forall w \in \Sigma^*, A(w) \triangleleft \nu$?
- **inclusion:** $\forall w \in \Sigma^*, A(w) \leq B(w)$?
- **equivalence:** $\forall w \in \Sigma^*, A(w) = B(w)$?

Decision Problems

$\perp < \nu$ for all values ν, $\triangleleft \in \{<, \leq\}$

- **functionality:** is A functional ?
  - PTIME for Sum and DSum, coNP for Ratio
- $(\triangleleft\nu)$-**emptiness:** $\exists w \in \Sigma^*, \perp < A(w) \triangleleft \nu$?
  - for functional WA: PTIME
- $(\triangleleft\nu)$-**universality:** $\forall w \in \Sigma^*, A(w) \triangleleft \nu$?
  - for functional WA: PSpace-c
- **inclusion:** $\forall w \in \Sigma^*, A(w) \leq B(w)$?
  - for functional WA: Decidable for Ratio, PSpace-c for Sum and DSum
- **equivalence:** $\forall w \in \Sigma^*, A(w) = B(w)$?
  - for functional WA: Decidable for Ratio, PSpace-c for Sum and DSum

Functionality Problem

- as mentioned in (Kirsten,Mäurer,05)
- squaring transducer techniques apply to Sum-automata (Béal, Carton, Prieur, Sakarovitch,03)
- also apply to discounted sum ...
- and in general to groups $(W, \otimes, e)$

Squaring Technique (Béal, Carton, Prieur, Sakarovitch,03)

(Adapted to groups)

- take the product of A with itself
- remove all **non** co-accessible pairs
- compute delays for states of A^2

$$\mathrm{Delay}(p,q) = \{V(r_1)^{-1} \otimes V(r_2) \mid \exists (p_0,q_0) \xrightarrow{(r_1,r_2)} (p,q)\}$$

- A is functional iff for all states (p,q) of A^2:
  1. $|\mathrm{Delay}(p,q)| \leq 1$ and,
  2. $\mathrm{Delay}(p,q) = \{e\}$ if p and q are final.

Squaring Technique (Béal, Carton, Prieur, Sakarovitch,03)

(Adapted to groups)

- take the product of A with itself
- remove all **non** co-accessible pairs
- compute delays for states of A^2

$$\mathrm{Delay}(p,q) = \{V(r_1)^{-1} \otimes V(r_2) \mid \exists (p_0,q_0) \xrightarrow{(r_1,r_2)} (p,q)\}$$

- A is functional iff for all states (p,q) of A^2:
  1. $|\mathrm{Delay}(p,q)| \leq 1$ and,
  2. $\mathrm{Delay}(p,q) = \{e\}$ if p and q are final.

It can be checked in PTime by unfolding A^2 while computing delays (stop whenever a pair has two delays).

Consequence: functionality is decidable in PTime for Sum-automata.

DSum as a group operation

DSum group

$(\mathbb{Q} \times (\mathbb{Q} - \{0\}), \otimes, e)$ where

- $(a, x) \otimes (b, y) = (\frac{a}{y} + b, xy)$
- $e = (0, 1)$
- $(a, x)^{-1} = (-xa, x^{-1})$

- Given $\lambda \in \mathbb{Q} \cap]0, 1[$,

$$
\begin{aligned}
& (a_1, \lambda) \otimes \cdots \otimes (a_n, \lambda) \\
= \quad & (\tfrac{1}{\lambda^{n-1}} a_1 + \tfrac{1}{\lambda^{n+2}} a_2 + \ldots a_n, \lambda^n) \\
= \quad & (\tfrac{DS(a_1 \ldots a_n)}{\lambda^n}, \lambda^n).
\end{aligned}
$$

- replace values a_i by (a_i, λ) in A
- A is functional $\Leftrightarrow$ the new WA is functional

Functionality of Ratio-Automata

- no delay notion
- squaring technique does not apply
- pumping

Lemma (Small witness property)

If a Ratio-automaton with n states is not functional, then there exists a string w s.t. $|w| < 4n^2$ with at least two different values.

Functionality of Ratio-Automata

- no delay notion
- squaring technique does not apply
- pumping

Lemma (Small witness property)

If a Ratio-automaton with n states is not functional, then there exists a string w s.t. $|w| < 4n^2$ with at least two different values.

- Similar to Schützenberger pumping argument to decide functionality of finite state transducers (with bound $3n^2$)
- for Ratio-automata, does not hold for the bound $3n^2$
- CoNP procedure
- PTime if weights are encoded in unary

Inclusion Problem $A_1 \leq A_2$ (for Sum and DSum)

Decidability known for functional Sum-WA (Krob, Litp, 94).

Product $A_1 \otimes A_2$

$$\left.\begin{array}{l} q_1 \xrightarrow{a:n_1} p_1 \in A \\ q_2 \xrightarrow{a:n_2} p_2 \in B \end{array}\right\} \Rightarrow (q_1, q_2) \xrightarrow{a:n_1-n_2} (p_1, p_2) \in A \otimes B$$

- $A_1 \not\leq A_2$ iff there exists a path from an initial to a final pair with sum (resp. Dsum) > 0.
- for Sum: use reversal-bounded counter machines, for instance
- for DSum: linear programming (Andersson, 06)

Inclusion Problem $A_1 \leq A_2$ (for Ratio)

Product $A_1 \otimes A_2$

$$\left.\begin{array}{l} q_1 \xrightarrow{a:(r_1,c_1)} p_1 \in A \\ q_2 \xrightarrow{a:(r_2,c_2)} p_2 \in B \end{array}\right\} \Rightarrow \quad (q_1, q_2) \xrightarrow{a:(r_1,r_2,c_1,c_2)} (p_1, p_2) \in A \otimes B$$

Procedure

- compute $\mathcal{P}$ the Parikh image of successful runs of $A_1 \otimes A_2$
- $\mathcal{P}$ is a semi-linear set of tuples $(x_{t_1}, \dots, x_{t_n})$ for all transitions t_i of $A_1 \otimes A_2$
- $A_1 \not\leq A_2$ iff there exists $(x_{t_i})_i \in \mathcal{P}$ such that

$$\frac{\sum_{t_i} x_{t_i}.r_1(t_i)}{\sum_{t_i} x_{t_i}.c_1(t_i)} > \frac{\sum_{t_i} x_{t_i}.r_2(t_i)}{\sum_{t_i} x_{t_i}.c_2(t_i)}$$

Inclusion Problem $A_1 \leq A_2$ (for Ratio)

$$\frac{\sum_{t_i} x_{t_i}.r_1(t_i)}{\sum_{t_i} x_{t_i}.c_1(t_i)} > \frac{\sum_{t_i} x_{t_i}.r_2(t_i)}{\sum_{t_i} x_{t_i}.c_2(t_i)}$$

is equivalent to

$$\sum_{t_i,t_j} x_{t_i}.x_{t_j}.r_1(t_i).c_2(t_j) > \sum_{t_i,t_j} x_{t_i}.x_{t_j}.r_2(t_i).c_1(t_j)$$

Inclusion Problem $A_1 \leq A_2$ (for Ratio)

$$\frac{\sum_{t_i} x_{t_i}.r_1(t_i)}{\sum_{t_i} x_{t_i}.c_1(t_i)} > \frac{\sum_{t_i} x_{t_i}.r_2(t_i)}{\sum_{t_i} x_{t_i}.c_2(t_i)}$$

is equivalent to

$$\sum_{t_i,t_j} x_{t_i}.x_{t_j}.r_1(t_i).c_2(t_j) > \sum_{t_i,t_j} x_{t_i}.x_{t_j}.r_2(t_i).c_1(t_j)$$

Quadratic Diophantine Equations

- x_{t_i} are variables with semi-linear constraint $\mathcal{P}$
- $r_1(t_i)$, $r_2(t_i)$, $c_1(t_i)$, $c_2(t_i)$ are constants.
- For **one quadratic diophantine equation** and a **set of semi-linear constraints**, existence of a solution is decidable (Grunewald, Segal, 04) (Wong, Krieg, Thomas, 06)

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in}) :

Player *out* (Σ_{out}) :

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in}) : i_1

Player *out* (Σ_{out}) :

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in}) : i_1

Player *out* (Σ_{out}) : o_1

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in}) : i_1 i_2

Player *out* (Σ_{out}) : o_1

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in})	:	i_1	i_2
Player *out* (Σ_{out})	:	o_1	o_2

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in})	:	i_1	i_2	i_3
Player *out* (Σ_{out})	:	o_1	o_2	

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in})	:	i_1	i_2	i_3
Player *out* (Σ_{out})	:	o_1	o_2	o_3

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in}) : i_1 i_2 i_3 i_4

Player *out* (Σ_{out}) : o_1 o_2 o_3

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4	i_5
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4	

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4	i_5
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4	o_5

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4	i_5	$\dots$	i_n
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4	o_5	$\dots$	o_n

Church Game (on Finite Strings)

Definition

- **turn-based** game between two players
- Player *in* (adversary) chooses input symbols in Σ_{in}
- Player *out* (protagonist) chooses output symbols in Σ_{out}

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4	i_5	$\dots$	i_n
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4	o_5	$\dots$	o_n

Finite String Restriction

Player *in* can stop the game. If it does not, then he loses.

Church Game (on Finite Strings)

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4	i_5	$\dots$	i_n
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4	o_5	$\dots$	o_n

Church Game (on Finite Strings)

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4	i_5	$\dots$	i_n
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4	o_5	$\dots$	o_n

Boolean Objective

- Winning objective for Player *out*: some NFA A over $\Sigma_{in} \times \Sigma_{out}$
- Player *out* wins if Player *in* never stops, or if it does, $(i_1, o_1) \dots (i_n, o_n) \in L(A)$.

Church Game (on Finite Strings)

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4	i_5	$\dots$	i_n
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4	o_5	$\dots$	o_n

Quantitative Objective

- Winning objective for Player *out*: some WA A over $\Sigma_{in} \times \Sigma_{out}$ and some threshold ν
- Player *out* wins if Player *in* never stops, or if it does, $A((i_1, o_1) \dots (i_n, o_n)) > \nu$.

Church Game (on Finite Strings)

Player in (Σ_{in}) : i_1 i_2 i_3 i_4 i_5 $\dots$ i_n

Player *out* (Σ_{out}) : o_1 o_2 o_3 o_4 o_5 $\dots$ o_n

Problem

$$\exists S_1 : \text{Histories} \to \Sigma_{out},\ \forall S_2 : \text{Histories} \to \Sigma_{in}$$

$$A(\text{outcome}(S_1, S_2)) > 0?$$

.

Church Game (on Finite Strings)

Player in (Σ_{in})	:	i_1	i_2	i_3	i_4	i_5	$\ldots$	i_n
Player *out* (Σ_{out})	:	o_1	o_2	o_3	o_4	o_5	$\ldots$	o_n

Problem

$$\exists S_1 : \text{Histories} \rightarrow \Sigma_{out},\ \forall S_2 : \text{Histories} \rightarrow \Sigma_{in}$$

$$A(\text{outcome}(S_1, S_2)) > 0?$$

.

Results (for functional)

- undecidable for Sum and Ratio, open for DSum
- decidable for sequential WA (Sum, DSum, Ratio)
- determinizability decidable for Sum (Kirsten,Mäurer,05) , DSum, open for Ratio

Ongoing work: extension to k-valued WA

Definition

- Given $k \in \mathbb{N}$, a WA A is k-valued if:

  for all strings $w \in \Sigma^* \cdot |\{A(r) \mid r \text{ accepting on } w\}| \leq k$

- Finite-valued if there exists k such that A is k-valued.

- Over semirings $(W, \oplus, \otimes)$, finite-valued WA are closed under $\oplus$.
- is k-valuedness decidable ? for $(\mathbb{Z}, max, +)$, for DSum ?
- what about inclusion ?

k-valued WA over $(\mathbb{Z}, max, +)$

- Given k, k-valuedness is decidable
- k-ambiguous WA $\leftrightarrow$ finite union of unambiguous WA
  - Klimann, Lombardy, Mairesse, Prieur, 04
- k-valued WA $\leftrightarrow$ finite union of unambiguous WA
  - holds true for string-to-string transducers
  - as shown in (Weber, 96), (Sakarovitch, de Souza, 10)
- inclusion reduces to the following problem:
  - Input: directed graph with edges labelled by tuples in $\mathbb{Z}^p$, two nodes s and t
  - Ouput: is there a path from s to t with positive sum on all dimensions?

k-valued DSum-Automata

- k-ambiguous WA $\leftrightarrow$ finite union of unambiguous WA
- inclusion reduces to the following problem:
  - Input: directed graph with edges labelled by tuples in $\mathbb{Z}^p$, two nodes s and t
  - Ouput: is there a path from s to t with positive **DSum** on all dimensions?
- (Chatterjee, Forejt, Wojtczak, 13): at least as difficult as the following open problem (in one dimension): is there a finite (infinite) path with DSum exactly 0 ?

$$q \xrightarrow{i} p \;\Rightarrow\; q \xrightarrow{(i,-i)} p$$

Conclusion

- functional WA have good (algorithmic) properties
- new techniques to handle ratio
- game problem undecidable for functional Sum-automata
- k-valued ?
- infinite strings ?