Implicit automata in λ-calculi III: affine planar string-to-string functions

Cécilia Pradic

Ian Price
countingishard.org

Swansea University

Mathematical Foundations of Programming Semantics
June 2024

Previous Work (STLC)

Theorem (Hillebrand & Kanellakis '96)

Let $L \subseteq \Sigma^\star$. *The following are equivalent:*

- *L can be defined by a simply typed* λ*-term of type* $\mathrm{Str}_\Sigma[\tau] \to \mathrm{Bool}$ *for some simple type* τ
- *L is a regular language*

Church Encodings

Definition (Bool)

$\mathrm{Bool} := \mathbb{o} \to \mathbb{o} \to \mathbb{o}$
$\mathsf{Church}(\mathrm{true}) := \lambda x.\, \lambda y.\, x$
$\mathsf{Church}(\mathrm{false}) := \lambda x.\, \lambda y.\, y$

Definition (Str_Σ)

Fix alphabet $\Sigma = \{a_1, \ldots, a_n\}$.

$$\mathrm{Str}_\Sigma[\tau] := \underbrace{(\tau \to \tau) \to \cdots \to (\tau \to \tau)}_{n \text{ times}} \to \tau \to \tau$$

$$\mathsf{Church}(w_1 \cdots w_m) := \lambda a_1.\, \cdots\, \lambda a_n.\, \lambda \varepsilon.\, w_1(\cdots(w_m\, \varepsilon))$$

$$\mathrm{append}_a = \lambda w.\, \lambda a_1.\, \cdots\, \lambda a_n.\, \lambda \varepsilon.\, w\, a_1\, \cdots\, a_n\, (a\, \varepsilon)$$

Proof Idea (Soundness Only)

Interpret λ in **FinSet**:

- $[\![\mathbb{o}]\!] = \{0, 1\}$
- $[\![\tau \to \sigma]\!] = [\![\tau]\!] \to [\![\sigma]\!]$

For each term $t : \mathrm{Str}_\Sigma[\tau] \to \mathrm{Bool}$, obtain DFA:

- $Q = [\![\mathrm{Str}_\Sigma[\tau]]\!]$
- $\delta(a) = [\![\mathrm{append}_a]\!]$
- $q_0 = [\![\epsilon]\!]$
- $F = \{q \in Q : [\![t]\!](q) = [\![\mathsf{Church}(\mathrm{true})]\!]\}$

Key Observation

$\delta(w)(q_0) = [\![\mathsf{Church}(w)]\!]$

Main Theorem

Theorem

The following are equivalent:

- *Affine string-to-string $\lambda\wp$ definable functions*
- *first-order string transductions*
- *planar reversible two-way finite transducers*

$\}$ *Nguyễn, Noûs, and Pradic '23*

Affine string-to-string definable functions

$\lambda\wp$ = Non-Commutative Affine Lambda Calculus

- ✓ $\lambda x.\, \lambda y.\, y$
- ✗ $\lambda x.\, \lambda y.\, x\, y\, y$
- ✗ $\lambda x.\, \lambda y.\, y\, x$

$$A, B \coloneqq \mathbb{o} \mid A \multimap B \mid A \to B$$

$$\mathrm{Str}_\Sigma[\tau] \coloneqq \underbrace{(\tau \multimap \tau) \to \cdots}_{|\Sigma| \text{ times}} \to \tau \to \tau$$

Definition (Affine Definable)

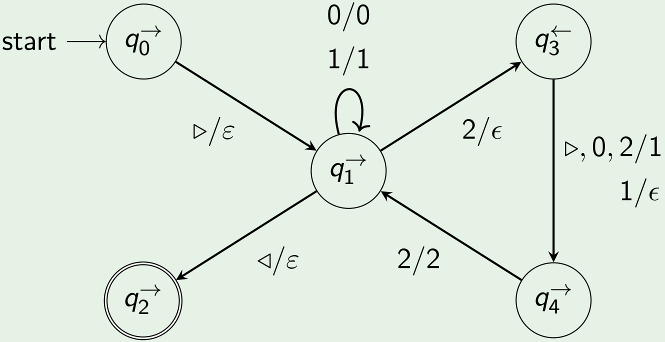
A function $f : \Sigma^* \to \Gamma^*$ is called *affine $\lambda\wp$-definable* when

- exists a purely affine type κ, and
- a λ-term $\mathtt{f} : \mathrm{Str}_\Sigma[\kappa] \multimap \mathrm{Str}_\Gamma$, s.t.
- $\forall s \in \Sigma^*, \mathrm{Church}(f(s)) =_{\beta\eta} \mathtt{f}\ \mathrm{Church}(s)$

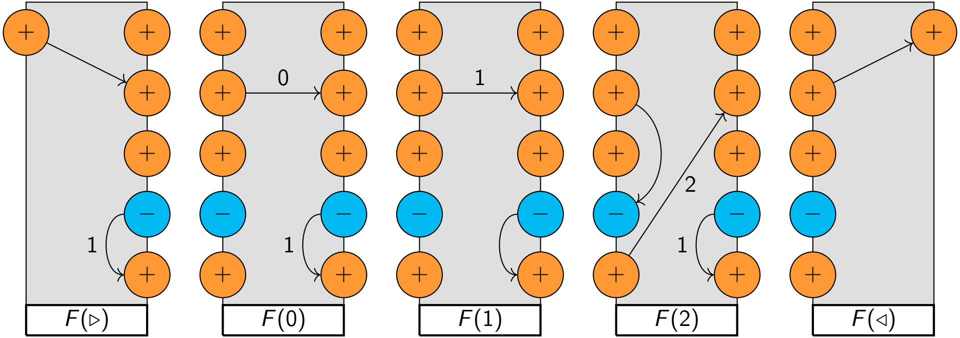
Two-Way Transducers

Example

The following 2DFT takes any string and ensures that every 2 is preceded by a 1 by adding 1s if necessary.



Two-Way Transducers (cont.)



Category of Words

Definition ($\mathbf{Shape}_\Sigma$)

For any finite alphabet Σ, there is a three object category $\boldsymbol{\mathit{Shape}}_\Sigma$ generated by the following finite graph,

$$\begin{tikzcd}
\mathrm{in} \arrow[r, "\triangleright"] & \mathrm{states} \arrow[loop above, "a\in\Sigma"] \arrow[r, "\triangleleft"] & \mathrm{out}
\end{tikzcd}$$

$$\begin{aligned}
\text{words over } \Sigma \quad &\cong \quad \text{morphisms } \mathrm{in} \to \mathrm{out} \\
\text{“abc”} \quad &\mapsto \quad \triangleright \,;\, a \,;\, b \,;\, c \,;\triangleleft
\end{aligned}$$

Automata as Functors

Definition (Automaton)

For any category $\mathcal{C}$ and objects $I, O \in \mathcal{C}$, a $(\mathcal{C}, I, O)$-*automaton with input alphabet* Σ

- a functor $\mathcal{A} : \mathbf{Shape}_\Sigma \to \mathcal{C}$, s.t.,
- $\mathcal{A}(\mathrm{in}) = I$, and
- $\mathcal{A}(\mathrm{out}) = O$.

Its semantics is the map $\Sigma^* \to [I, O]_\mathcal{C}$ given by $w \mapsto \mathcal{A}(\triangleright) \mathbin{;} \mathcal{A}(w) \mathbin{;} \mathcal{A}(\triangleleft)$.

Definition (DFA)

A *deterministic finite automaton* with input alphabet Σ is a $(\mathbf{Set}, \{\bullet\}, \{\mathrm{true}, \mathrm{false}\})$-automaton.

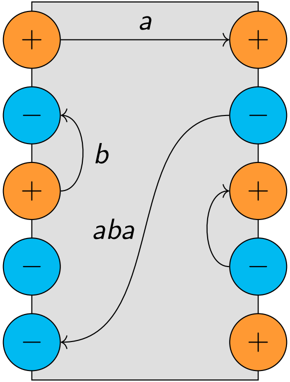
Transition Diagrams

Compared with DFAs, 2RFTs have more structure:

- We can go forwards and backwards along the tape
- We need some way to "output" strings
- We require reversibility & planarity

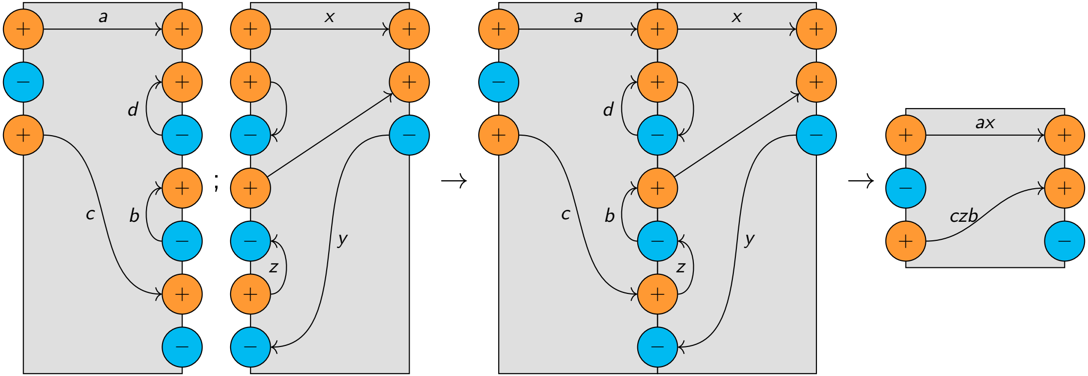
This is solved by introducing a new category of "transition diagrams" **TransDiag**.

Objects & Morphisms



Composition

Glue morphisms together and concatenate strings



2PRFTs as a Functor

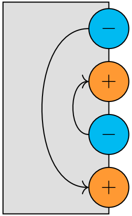
Definition (2PRFT)

A *two-way planar reversible transducer* $\mathcal{T}$ with input alphabet Σ and output alphabet Γ is a $(\mathbf{TransDiag}_\Gamma, \varepsilon, +-)$-automaton with input alphabet Σ.

Category Round-Up

TransDiag is very Nice™

- Strict Monoidal
- **Poset**$_\perp$-enriched
- Pivotal Category (dualizing structure)
- Suitable for interpreting $\lambda_\wp$



Interpreting $\lambda\wp$ in **TransDiag**

$$\frac{x \text{ a variable of } \underline{\Gamma}}{\underline{\Gamma}; \Delta \vdash x : \tau} \quad \longmapsto \quad [\![x]\!] \circ \bot_{[\![\Delta]\!]} : [\![\Delta]\!] \to [\![\tau]\!]$$

$$\frac{}{\underline{\Gamma}; \Delta, x : \tau, \Delta' \vdash x : \tau} \quad \longmapsto \quad \bot_{[\![\Delta]\!]} \otimes \mathrm{id}_{[\![\tau]\!]} \otimes \bot_{[\![\Delta']\!]} : [\![\Delta]\!] \otimes [\![\tau]\!] \otimes [\![\Delta']\!] \to [\![\tau]\!]$$

$$\frac{\underline{\Gamma}; \Delta, x : \tau \vdash t : \sigma}{\underline{\Gamma}; \Delta \vdash \lambda x.t : \tau \multimap \sigma} \quad \longmapsto \quad \frac{[\![t]\!] : [\![\Delta]\!] \otimes [\![\tau]\!] \to [\![\sigma]\!]}{\Lambda_{[\![\Delta]\!],[\![\tau]\!],[\![\sigma]\!]}([\![t]\!]) : [\![\Delta]\!] \to [\![\tau]\!] \multimap [\![\sigma]\!]}$$

$$\frac{\underline{\Gamma}; \Delta \vdash t : \tau \multimap \sigma \qquad \underline{\Gamma}; \Delta' \vdash u : \tau}{\underline{\Gamma}; \Delta, \Delta' \vdash t\ u : \sigma} \quad \longmapsto \quad \frac{[\![t]\!] : [\![\Delta]\!] \to [\![\tau]\!] \multimap [\![\sigma]\!] \qquad [\![u]\!] : [\![\Delta']\!] \to [\![\tau]\!]}{\mathrm{ev}_{[\![\tau]\!],[\![\sigma]\!]} \circ ([\![t]\!] \otimes [\![u]\!]) : [\![\Delta]\!] \otimes [\![\Delta']\!] \to [\![\sigma]\!]}$$

Interpreting Reductions

Lemma

- *If* $t \to_\eta u$*, then* $[\![t]\!] = [\![u]\!]$*.*
- *If* $t \to_\beta u$*, then* $[\![t]\!] \geq [\![u]\!]$*.*

Corollary

If t *has a normal form* t_{NF}*, then* $[\![t_{\mathrm{NF}}]\!] \leq [\![t]\!]$*.*

Main Theorem

Theorem

The following are equivalent:

1. *Affine string-to-string* $\lambda\wp$ *definable functions*
2. *first-order string transductions*
3. *planar reversible two-way finite transducers*

We turn to the proof that (1) implies (3).

Proof of Soundness

Step 1. Apply the following lemma to obtain o, d_i, d_ϵ.

Lemma

Let $\Sigma = \{a_1, \dots, a_n\}$ *and* $\Gamma = \{b_1, \dots, b_k\}$ *be alphabets.*
Up to $\beta\eta$*-equivalence, every term of type* $\mathrm{Str}_\Sigma[\kappa] \multimap \mathrm{Str}_\Gamma$ *is of the shape*

$$\lambda s.\, \lambda b_1.\, \dots\, \lambda b_k.\, \lambda\epsilon.\, o\,(s\ d_1\ \dots\ d_n\ d_\epsilon)$$

where o, d_ϵ *and the* d_i*s have typing derivations*

$$\underline{\Gamma}; \cdot \vdash o : \kappa \multimap \mathbb{O} \qquad \underline{\Gamma}; \cdot \vdash d_i : \kappa \multimap \kappa \qquad \underline{\Gamma}; \cdot \vdash d_\epsilon : \kappa$$

Proof of Soundness (cont.)

Step 2. Apply the interpretation to those terms

$$[\![d_a]\!] : \mathsf{I} \to [\![\kappa]\!] \multimap [\![\kappa]\!] \qquad [\![o]\!] : \mathsf{I} \to [\![\kappa]\!] \multimap + - \qquad [\![d_\epsilon]\!] : \mathsf{I} \to [\![\kappa]\!]$$

Step 3. Define 2PRFT

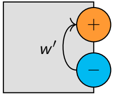
$$\mathcal{T}(a) \;=\; \Lambda^{-1}_{\mathsf{I},[\![\kappa]\!],[\![\kappa]\!]}([\![d_a]\!]) \qquad \mathcal{T}(\triangleleft) \;=\; \Lambda^{-1}_{\mathsf{I},[\![\kappa]\!],[\![\mathbb{o}]\!]}([\![o]\!]) \qquad \mathcal{T}(\triangleright) = [\![d_\epsilon]\!]$$

Step 4. Do a little calculation to check this computes the same function

Proof of Soundness (cont.)

For input word $w = w_1 \dots w_n \in \Sigma^*$, let $f(w) = w'$.

$$\begin{aligned}
\mathcal{T}(\triangleright w \triangleleft) &= \mathcal{T}(\triangleleft) \circ \mathcal{T}(w_n) \circ \dots \circ \mathcal{T}(w_1) \circ \mathcal{T}(\triangleright) \\
&= \dots \\
&= [\![o\ (d_{w_n}\ \dots (d_{w_1}\ d_\epsilon) \dots)]\!] \\
&\geq [\![\mathrm{Church}(w')]\!] \\
&=
\end{aligned}$$



$\dots$ but $\geq$ is really $=$ because diagram is maximal. ☐

Wrapping Up

Other direction: apply Krone-Rhodes decomposition theorem

Extensions

- Dropping Planarity: first-order → regular
- $\mathrm{Str}_{\Sigma}[\kappa] \to \mathrm{Str}_{\Gamma}$: first-order comparison-free functions (?)

Broader Picture

$\mathrm{Str}_\Sigma[A] \multimap \mathrm{Bool}$ with A linear (adapted as needed):

λ-calculus	languages	status
simply typed	regular	✓ [Hillebrand & Kanellakis '96]
linear or affine	regular	✓
non-commutative linear or affine	star-free	✓

$\mathrm{Str}_\Gamma[A] \multimap \mathrm{Str}_\Sigma$ with A affine (adapted as needed):

λ-calculus	transducers	status
linear (without additives)	weird (?)	?
affine	regular functions	✓
non-commutative affine	first-order regular fn.	✓
linear/affine with additives	regular functions	✓
parsimonious	polyregular	??
simply typed	variant of CPDA???	???

References

- Colcombet and Petrișan, “Automata Minimization: a Functorial Approach”
- Hillebrand and Kanellakis, “On the expressive power of simply typed and let-polymorphic lambda calculi”
- Nguyễn and Pradic, “Implicit Automata in typed λ-calculi I: Aperiodicity in a Non-Commutative Logic”
- Nguyễn, Noûs, and Pradic, “Implicit Automata in typed λ-calculi II: streaming transducers vs categorical semantics”
- Nguyễn, Noûs, and Pradic, “Two-way automata and transducers with planar behaviours are aperiodic”

Thank You!
Any Questions?