Tier-Based Strictly Local Stringsets

Perspectives from Model and Automata Theory

Dakotah Lambert[1] James Rogers[2]

[1]Stony Brook University
Department of Linguistics

[2]Earlham College
Department of Computer Science

SCiL
New Orleans, January 2–5, 2020

Outline

Generalizing local relationships

Characterizing TSL
- A general characterization
- A simplification for strings

Automata and operations

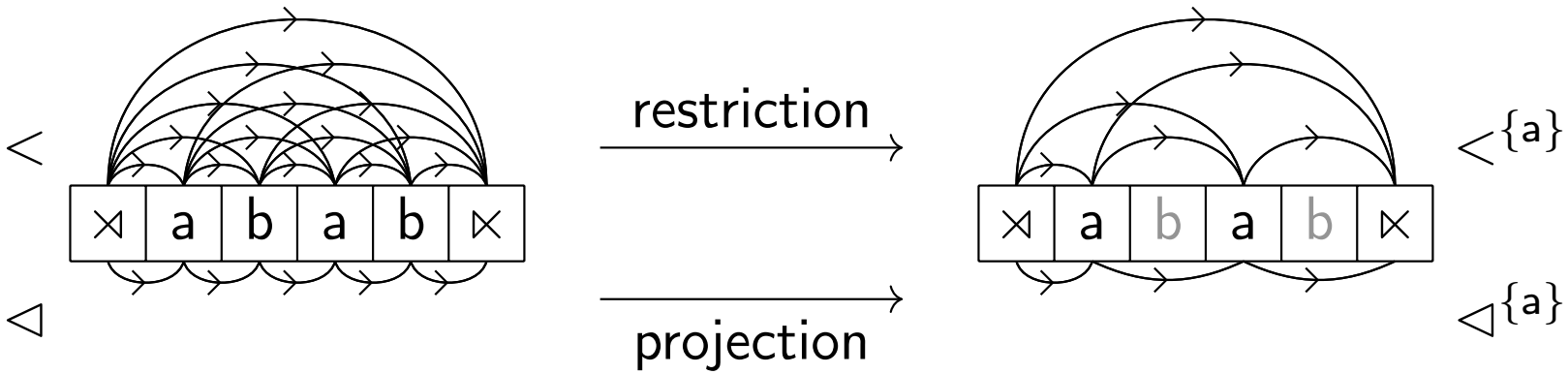
A model-theoretic view: relational word models

Relevant information

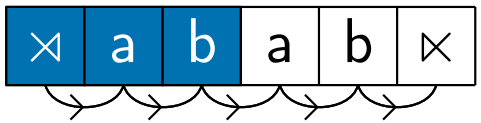
- Sequence of positions in $\rtimes w \ltimes$: D_w
- Set of unary symbol relations: σ_w, $\rtimes_w$, and $\ltimes_w$
- Arbitrary relations: R_i

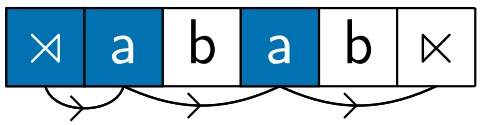
$$\mathcal{M}_{\Sigma}^{R_i}(w) \triangleq \langle D_w, \sigma_w, \rtimes_w, \ltimes_w, R_i \rangle_{\sigma \in \Sigma}$$

Some relations: (projective) precedence and adjacency



Windows and factors





Formal (inductive) definition

For a relation R of arity a:

$$W_a^R(x_1) \triangleq \{\langle x_1, \ldots, x_a\rangle : (x_1, \ldots, x_a) \in R\}$$

$$W_{i+1}^R(x_1) \triangleq \{\langle x_1, \ldots, x_{i+1}\rangle : \langle x_1, \ldots, x_i\rangle \in W_i^R(x_1) \text{ and } (x_{i-a+2}, \ldots, x_{i+1}) \in R\}$$

Formulae in a propositional logic

Intent	Expression
f occurs	f
a and b	$a \wedge b$
a or b	$a \vee b$
not a	$\neg a$

$\{f_i\}$ means $\bigvee(f_i)$

Classification by definability

Definition (Testable)

Definable with a propositional formula

Definition (Testable in the strict sense)

Testable where formula is of the form $\neg \overline{G}$

	$\lhd$	$\lhd^{\tau}$	$<$
Testable	LT	TLT	PT
Strict-Testable	SL	TSL	SP

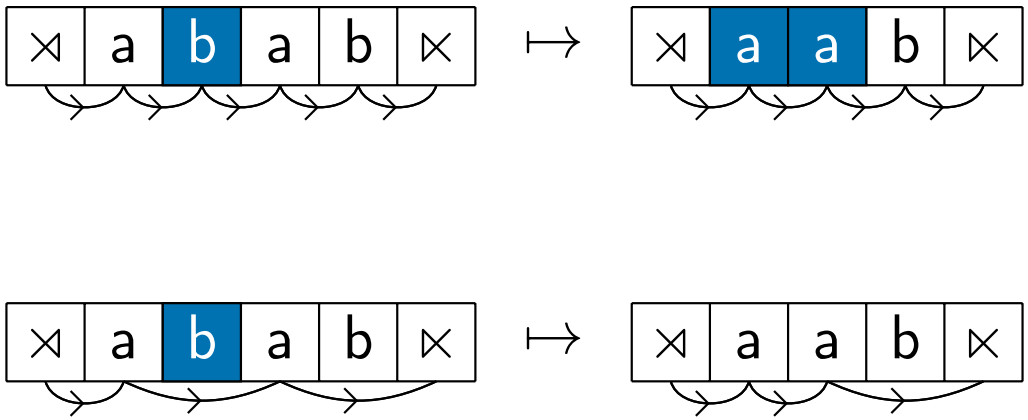
Mixing relations makes things more complicated.

Conservativity and realizability

$$f : X^n \to X$$

- No new factors
- Still a valid structure (realizable)

Conservativity depends on factor width and relation

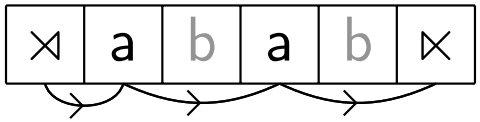


Conservative under $\lhd^{\{a\}}$ but not under $\lhd$
(if $k \geq 2$)

A general characterization

Strictly testable if and only if
closed under **all** conservative operations

Insertion and deletion of non-tier symbols



- $\lhd^{\tau}$ is indifferent to symbols outside τ.
- Inserting or deleting such symbols is conservative.
- All TSL stringsets are closed under those operations.

Substitution of suffixes

Definition (Suffix substitution)

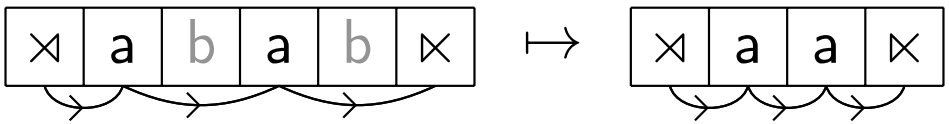
For x of length $k-1$, axc and dxf produce axf.

SL: closure under suffix substitution.
What about TSL?

A relational view on projection

Definition (Projection to τ)

$\pi_\tau(w)$: the longest factor of w under $\lhd^\tau$



If L is TSL^τ, then $\pi_\tau(L) \subseteq L$.

Preprojective substitution of suffixes

Definition

For x of length $k-1$,
if $\pi_\tau(w_1) = axc$, $\pi_\tau(w_2) = dxf$, and $\pi_\tau(w_3) = axf$
then w_1 and w_2 produce w_3.

Project to τ, substitute suffixes, invert the projection

Conservative

Necessary and sufficient

Preprojective suffix substitution closure means $\pi_\tau(L)$ is SL.

Deletion and insertion closure allow
free projection and inverse-projection.

Combination implies TSL.

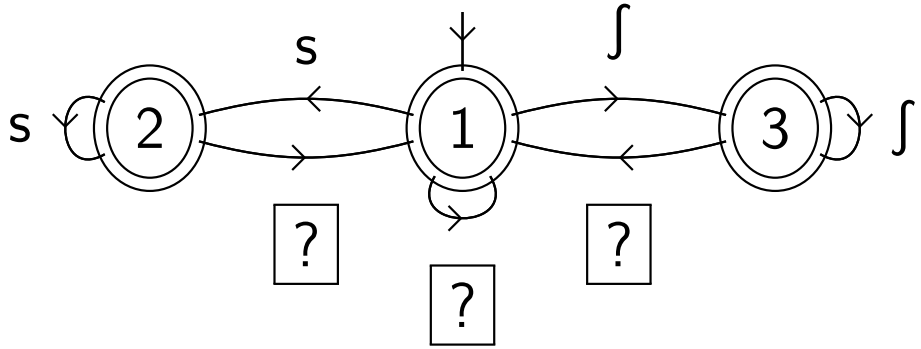
Proving something is not TSL

Find a subset of the tier alphabet:
things not freely deletable or insertable

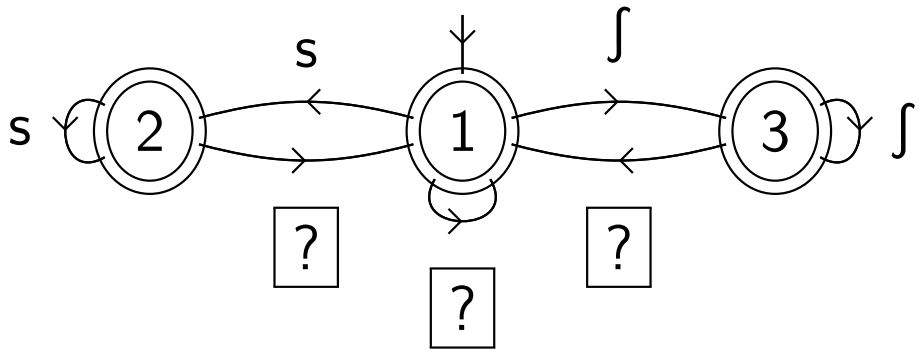
Find strings over this subset
that disprove suffix substitution closure

Local assimilation

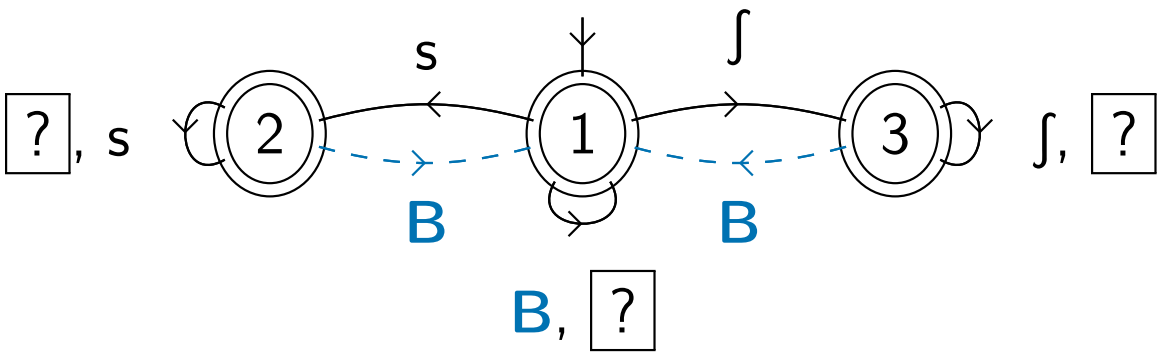
$$\neg s\int \wedge \neg \int s$$



Long-distance assimilation, with or without blockers



becomes



Determining if an automaton recognizes TSL

$$\bar{\tau} = \{\mathsf{L}\}$$

Projection is SL_2: $\neg\{\rtimes\mathsf{H}, \mathsf{H}\acute{\mathsf{H}}, \acute{\mathsf{H}}\acute{\mathsf{H}}\}$

$$\mathsf{TSL}_2^{\{\mathsf{H},\acute{\mathsf{H}}\}}$$

Conclusions

- Relation that abstracts away from projection
- Characterization of TSL
- Constructing TSL automata (and MTSL, TLT, etc.)
- Determining if Regular stringset is TSL