

Infinite state automata

Kamal Lodaya

Institute of Mathematical Sciences,
Chennai

Finite state automata

- What is a finite state automaton?

Finite state automata

- What is a finite state automaton?
- A **transition graph** (directed, edge-labelled) with two kinds of marked vertices

$$\mathcal{G} = (V, \{E_a \mid a \in A\}, I, F); \quad E_a \subseteq V \times V; \quad I, F \subseteq V$$

V finite, A finite

Finite state automata

- What is a finite state automaton?
- A **transition graph** (directed, edge-labelled) with two kinds of marked vertices

$$\mathcal{G} = (V, \{E_a \mid a \in A\}, I, F); \quad E_a \subseteq V \times V; \quad I, F \subseteq V$$

V finite, A finite

- This graph is said to **accept** words which are in

$$\text{Traces}(\mathcal{G}) = \{w \mid I \xrightarrow{E_w} F\}$$

$$E_{a_1 a_2 \dots a_n} = E_{a_1} \circ E_{a_2} \circ \dots \circ E_{a_n}$$

Verification: an automatic view

- Need to know temporal/monadic second order (MSO) logic ?

Verification: an automatic view

- Need to know temporal/monadic second order (MSO) logic ?
- First idea (Büchi ZML60): Satisfiability of logic reduces to emptiness of accepted language of automaton

Verification: an automatic view

- Need to know temporal/monadic second order (MSO) logic ?
- First idea (Büchi ZML60): Satisfiability of logic reduces to emptiness of accepted language of automaton
- Second idea: Emptiness reduces to reachability, e.g. $pre^*(C)$ or $post^*(C)$ for $C \subseteq V$

Verification: an automatic view

- Need to know temporal/monadic second order (MSO) logic ?
- First idea (Büchi ZML60): Satisfiability of logic reduces to emptiness of accepted language of automaton
- Second idea: Emptiness reduces to reachability, e.g. $pre^*(C)$ or $post^*(C)$ for $C \subseteq V$
- On FSA, there are efficient algorithms for reachability and emptiness

Infinite transition graphs

- What is an infinite state automaton?

Infinite transition graphs

- What is an infinite state automaton?
- A transition graph \mathcal{G} with V infinite, A finite

Infinite transition graphs

- What is an infinite state automaton?
- A transition graph \mathcal{G} with V infinite, A finite
- This is too general
Allows configuration graphs of Turing machines
 V need not even be recursively enumerable

Infinite transition graphs

- What is an infinite state automaton?
- A transition graph \mathcal{G} with V infinite, A finite
- This is too general
 - Allows configuration graphs of Turing machines
 - V need not even be recursively enumerable
- How is this \mathcal{G} to be presented?
 - How do we run algorithms on it?

Regular transition graphs

- Vertices are named from (another) finite alphabet B
- $V \subseteq B^*$; $I, F \subseteq V$; V, I, F *regular*
- This is sufficient even for Turing machines
- **Question:** How do we present E_a regularly, so that we have algorithms for emptiness and reachability?

End-regular graphs

Consider rooted graphs \mathcal{G} which have bounded degree. Fix a vertex v whose distance (length of shortest path) from the root is n .

$\mathcal{G}(v)$ is the subgraph of \mathcal{G} which has all vertices at distance less than n removed, rooted at vertex v .

$v \approx v'$ if the subgraphs $\mathcal{G}(v)$ and $\mathcal{G}(v')$ are isomorphic.

Definition 1 (Muller, Schupp TCS85) *A rooted, bounded degree regular transition graph is said to be **end-regular** if the equivalence \approx is of finite index.*

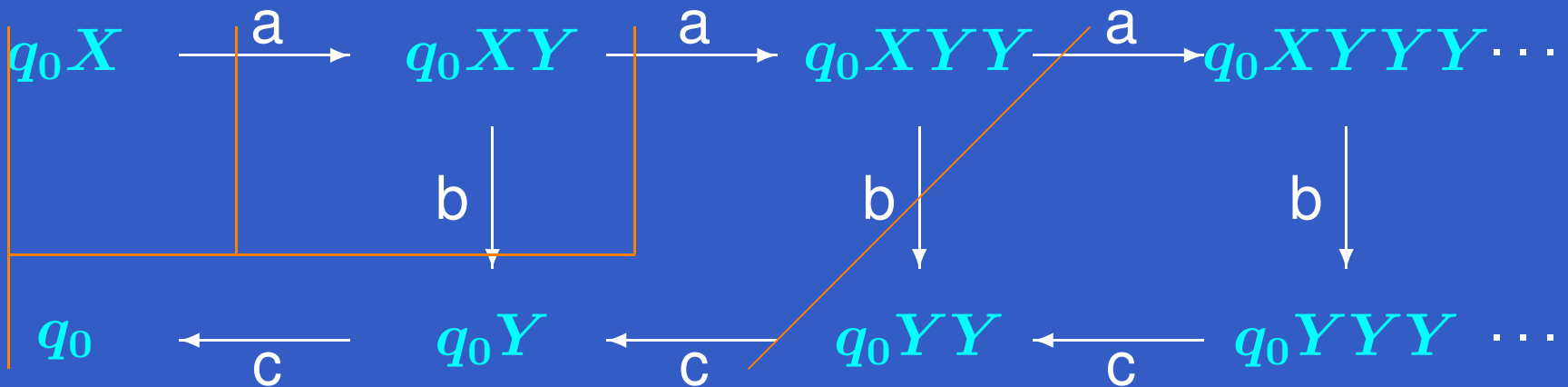
Pushdown transition graphs

Example: PDA for $\{a^nbc^n \mid n \geq 0\}$

$$\delta(q_0, a, X) = (XY, q_0)$$

$$\delta(q_0, b, X) = (\varepsilon, q_0)$$

$$\delta(q_0, c, Y) = (\varepsilon, q_0)$$



Muller and Schupp's theorem

- **Theorem 2 (Muller, Schupp TCS85)** *End-regular graphs are the transition graphs of PDA.*
- **Corollary 3** *The traces of end-regular graphs are the context-free languages.*
[Muller and Schupp called them **context-free graphs**, which have a different meaning in the graph grammar community.]

Verification

- Emptiness of accepted language is decidable for PDA
- **Theorem 4 (Büchi AML64)** *The set of configurations $V \subseteq QB^*$ of a PDA with states Q and stack alphabet B is regular.*
- **Theorem 5 (Esparza et al CAV00)** *Given a PDG $\mathcal{G} = (V, \{E_a \mid a \in A\})$ and regular $C \subseteq V$, there are efficient algorithms to compute $pre^*(C)$ and $post^*(C)$.*

A generalization

Definition 6 (Engelfriet, van Oostrom JCSS96) A *regular path description* is a regular transition graph with $E_a \subseteq \overline{B}^* B^*$ (or $B^* \overline{B} B^*$), E_a *regular*

(Caucal ICALP96/TCS03 calls this an inverse rational substitution followed by a rational restriction of the complete binary tree.)

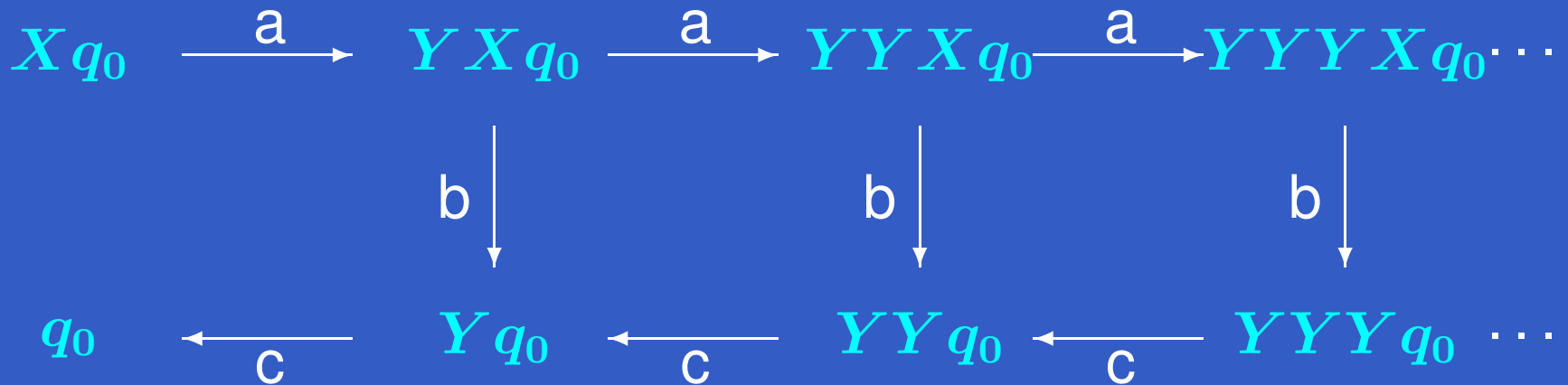
Regular path descriptions

Example: $V = Y^* X q_0 \cup Y^* q_0$

$$E_a = \overline{q_0 X} Y X q_0$$

$$E_b = \overline{q_0 X} q_0$$

$$E_c = \overline{q_0 Y} q_0$$



Graphs generated by prefix rewriting

- **Definition 7 (Büchi AML64; Caucal ICALP96/TCS03)**
*A **prefix-recognizable graph** is a regular transition graph with $E_a \subseteq (U_1 \times U_2)U_3$ where U_1, U_2, U_3 are regular.*
(These may be of unbounded degree.)
- **Theorem 8 (Caucal ICALP96/TCS03)** *The prefix-recognizable graphs are the regular transition graphs with regular path descriptions.*
(Caucal calls them **prefix-recognizable** graphs.)

Pushdown transition graphs again

Theorem 9 (Stirling 00) *The prefix-recognizable graphs are the transition graphs of PDA with ϵ -moves.*

Corollary 10 *The traces of these graphs are the context-free languages.*

Corollary 11 *The bounded degree prefix-recognizable graphs are the end-regular graphs.*

Verification

To summarize: we can do verification on

- FIN = finite transition graphs = transition graphs of FSA
= finite PRG
- ERG = end-regular graphs = transition graphs of PDA
= bounded-degree PRG
- PRG = prefix-recognizable graphs = regular transition
graphs with regular path descriptions = transition
graphs of PDA with ϵ -moves

Theorem 12 (Eng, van O Jcss97; Barthelmann 97)

*Regular transition graphs with regular path descriptions
are those which are MSO-interpretable in the binary tree.*

Generalizing further

- **Definition 13 (Morvan FOSSACS00)** A *rational graph* is a regular transition graph where $E_a \subseteq B^* \times B^*$ is a rational relation.
- **Rational relations** (Elgot and Mezei IBMJ65) built up from \emptyset , $\{(b, \varepsilon) \mid b \in B\}$ and $\{(\varepsilon, b) \mid b \in B\}$ using $\circ, \cup, *$

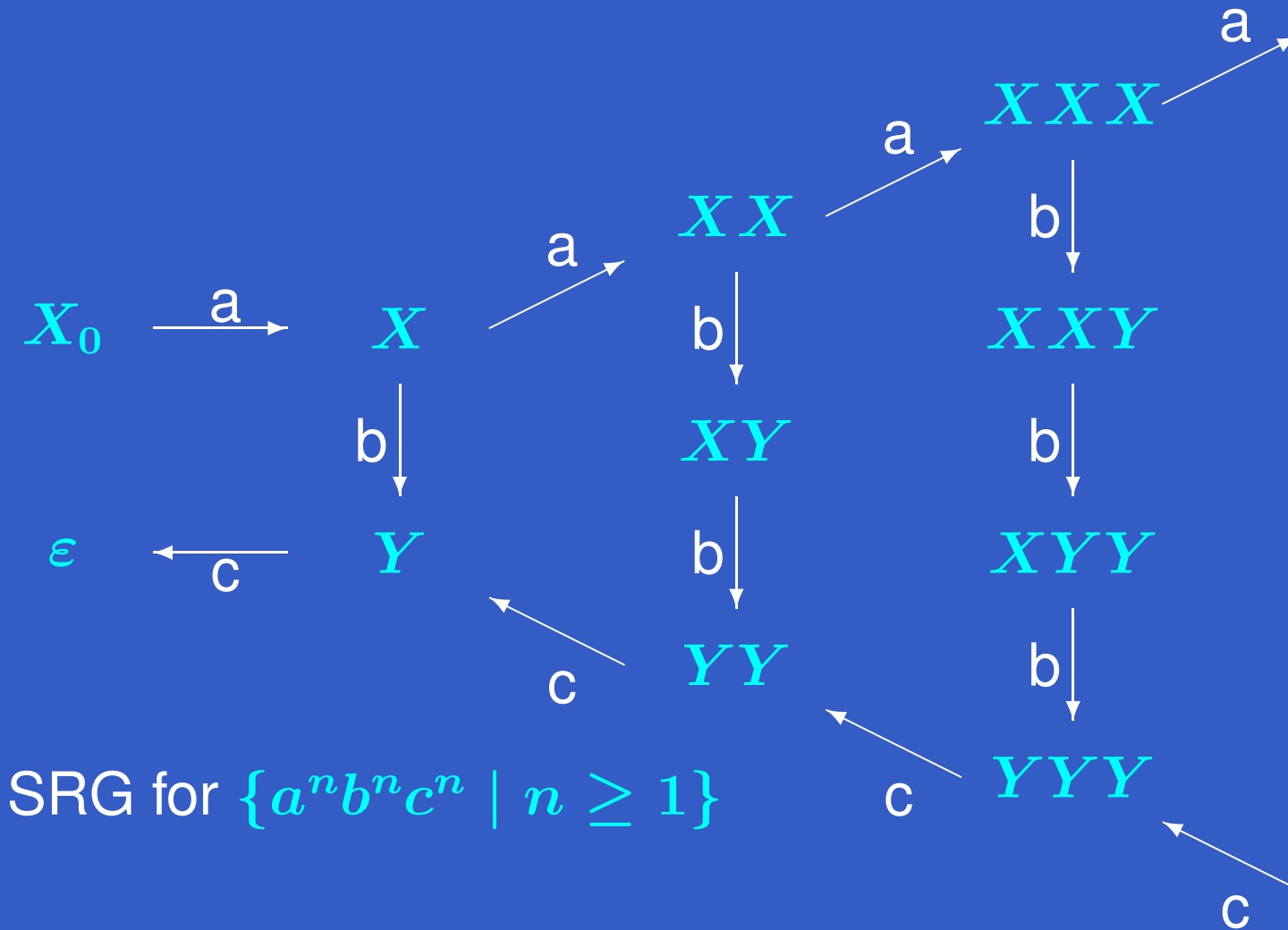
Rational relations

- **Rational transducer** (automaton): works on an input $(u, v) \in B^* \times B^*$ asynchronously
Sufficient to consider transition function
$$\delta : Q \times ((B \times \{\epsilon\}) \cup (\{\epsilon\} \times B)) \rightarrow \wp(Q)$$
- **Theorem 14 (Elgot, Mezei IBMJ65)** *A word relation is rational iff it can be accepted by a rational transducer.*
- (A **generalized sequential machine** has $\delta : Q \times B \times (B \cup \{\epsilon\}) \rightarrow \wp(Q)$; this is strictly weaker.)

Synchronized rational graphs

- **Definition 15 (Büchi ZML60)** A *synchronized rational graph* (aka sequential/automatic graph) is a regular transition graph where $E_a \subseteq (B \times B)^*((B^* \times \{\epsilon\}) \cup (\{\epsilon\} \times B^*))$ is a synchronized rational relation.
- **Synchronized rational relations**: accepted by sequential transducers, working on input $(u, v) \in B^* \times B^*$ with a bounded delay
Need transition functions $\delta : Q \times (B \times B) \rightarrow Q$,
 $\delta' : Q \times (B \times \{\epsilon\}) \rightarrow Q$, $\delta'' : Q \times (\{\epsilon\} \times B) \rightarrow Q$
- Sufficient for configuration graphs of Turing machines, hence reachability is undecidable

Example



Context-sensitive languages

- **Theorem 16 (Morvan FOSSACS00)** *The traces of rational graphs are context-sensitive languages.*
- **Theorem 17 (Morvan, Stirling MFCS01)** *Every context-sensitive language is the trace of a rational graph.*
- **Open question (Kuroda INFCONTR64):** What about deterministic CSLs?

Morvan's theorem

Wlg, \mathcal{G} has a single source state ' i ' in I and a single sink ' f ' in F . Suppose \mathcal{G} accepts the word $a_1 a_2 \dots a_n \in A^*$. Let $M_a = (Q_a, \delta_a, s_a, T_a)$ be the rational transducers for E_a over $(B \times \{\varepsilon\}) \cup (\{\varepsilon\} \times B)$,

$$A' = A \cup \{\vdash, \dashv, \text{blank}\}, B' = B \cup \{i, f, \varepsilon\},$$

$$Q' = \bigcup_a Q_a$$

An LBM with alphabet $A' \cup B' \cup Q'$ takes the input

$$\vdash \underbrace{a_1 a_2 \dots a_n}_{q_0} \dashv \underbrace{\text{blanks}}_{n+2} \text{ in } \vdash A^* \dashv \text{blank}^*$$

and rewrites it to

$$\vdash \underbrace{s_{a_1}}_{q_1} i s_{a_2} \varepsilon \dots s_{a_n} \varepsilon \dashv \varepsilon \text{ in } \vdash Q'_{a_1} B' \dots Q'_{a_n} B' \dashv B'$$

Simulation of rational transducers

In general, the configuration is a string in

$\vdash \dots Q'_a B' Q'_b B' \dots \dashv B'$ like

$$\vdash \dots \underbrace{p_a}_{q_1} \alpha p_b \beta \dots \dashv \gamma$$

which is rewritten to

$$\vdash \dots p'_a \varepsilon \underbrace{p_b}_{q_1} \beta \dots \dashv \gamma$$

if M_a goes from p_a to p'_a on (α, ε) , $\alpha \in B'$ (consumes input)

Simulation of rational transducers

The configuration

$$\vdash \dots \underbrace{p_a}_{q_1} \alpha p_b \varepsilon \dots \dashv \gamma$$

is rewritten to

$$\vdash \dots p'_a \alpha \underbrace{p_b}_{q_1} \beta \dots \dashv \gamma$$

if M_a goes from p_a to p'_a on (ε, β) , $\beta \in B'$ (produces output)

Simulation of rational transducers

- The LBM can also move left nondeterministically to its simulation of another transducer
- The LBM checks for success by seeing if it is at a configuration

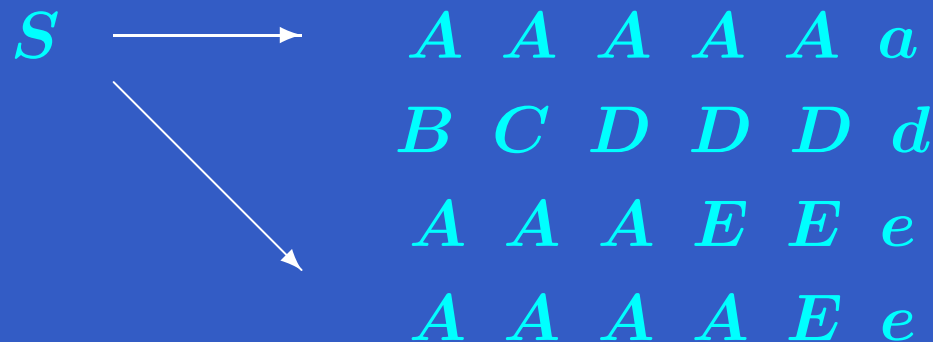
$$\boxed{\vdash t_{a_1} \varepsilon t_{a_2} \varepsilon \dots t_{a_n} \varepsilon \dashv f} \text{ in } \vdash T_{a_1} \varepsilon \dots T_{a_n} \varepsilon \dashv f$$

This means that all the transducers have consumed all their input and yielded a path

$$i E_{a_1} v_1 E_{a_2} v_2 \dots v_{n-1} E_{a_n} f, v_i \in V \subseteq B^*$$

Penttonen normal form

Theorem 18 (Penttonen INFCONTROL74) *There is a linear language L such that every CSL is derived from L by a 2-left-sensitive length-preserving derivation.*



linear 2-left-sensitive length-preserving

$ABAA \in L$

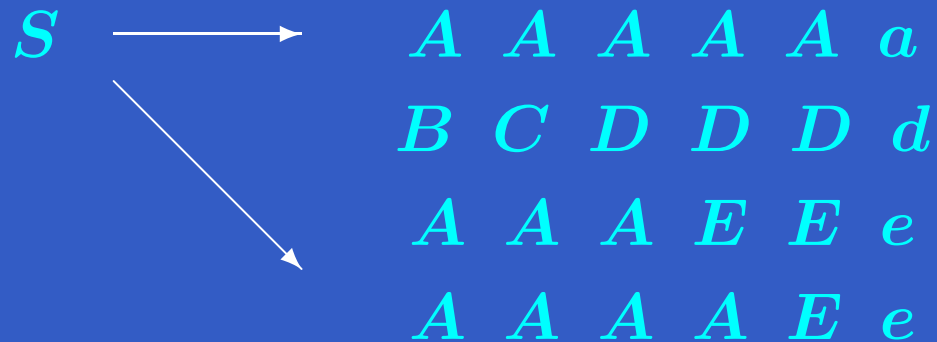
$G = \{AB \rightarrow AC, AC \rightarrow AD, DA \rightarrow DE, EA \rightarrow EE\}$

The Morvan-Stirling-Rispal theorem

Theorem 17 (Morvan, Stirling MFCS01) *Every context-sensitive language is the trace of a rational graph.*

Theorem 19 (Rispal INFINITY02) *Every context-sensitive language is the trace of a synchronized rational graph.*

The Morvan-Stirling-Rispal theorem

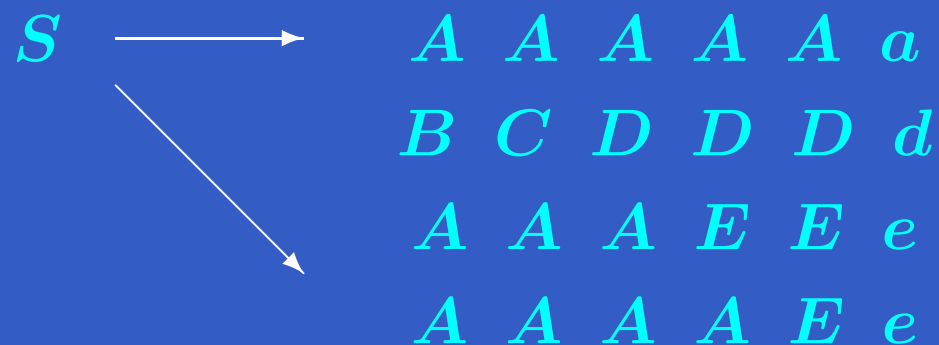


is represented by the synchronized rational relations

$$\begin{aligned}
 & [AAAAA]B E_a [BCDDDD] \\
 & [BCDDDD]A E_d [AAAEAE] \\
 & [AAAEAE]A E_e [AAAAAE] \\
 & [AAAAAE] E_e \varepsilon
 \end{aligned}$$

The Morvan-Stirling-Rispal theorem

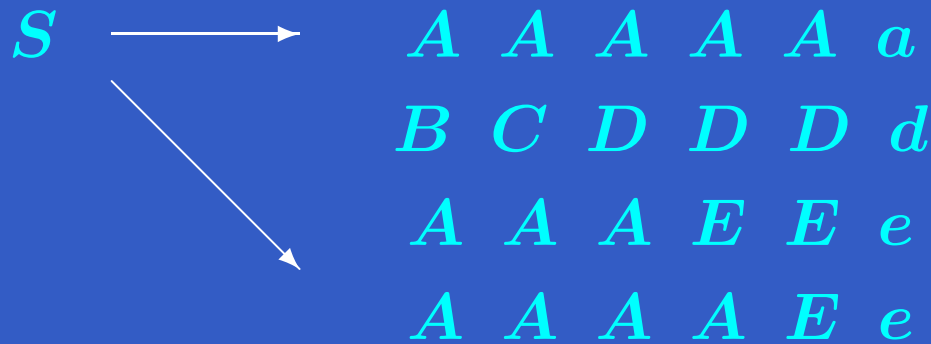
Incorporating the linear grammar



is represented by the synchronized rational relations

$$\begin{aligned}
 & S E_a [BCDDD]AW_1 \\
 & [BCDDD]AW_1 E_d [AAAE E]AW_2 \\
 & [AAAE E]AW_2 E_e [AAAAE]W_3 \\
 & [AAAAE]W_3 E_e \varepsilon
 \end{aligned}$$

Picture languages



It looks like if a picture language P represents 2-left-sensitive length-preserving derivations of a CSG, then $frontier(P)$ is a CSL.

Online tessellation automata

Definition 20 (Inoue, Nakamura INFSci77) An *online tessellation automaton* (Q, δ, I, F) has Q, I, F as usual and $\delta \subseteq (Q \times Q) \times B \times Q$.

```
# # # # # # #
# D A A A a #
# B D A A a #
# B B D A a #
# B B B D d #
# # # # # # #
```

Theorem 21 (Latteux, Simplot INFComput97) If a picture language P is accepted by an online tessellation automaton, then $\text{frontier}(P)$ is a CSL.

Frontier languages

- **Theorem 22 (Latteux, Simplot INFComput97)** *For every CSL L there is a picture language P accepted by an OTA, such that $L = \text{frontier}(P)$.*
- Generalizes the result that context-free languages are the frontiers of languages accepted by finite tree automata.
- **Open question (Kuroda INFContr64):** What about deterministic CSLs?

Verification

- All these context-sensitive languages may be theoretically very nice ...

Verification

- All these context-sensitive languages may be theoretically very nice ...
- But reachability/emptiness are undecidable for synchronized rational graphs, rational graphs, OTA

Verification

- All these context-sensitive languages may be theoretically very nice ...
- But reachability/emptiness are undecidable for synchronized rational graphs, rational graphs, OTA
- Are there classes of graphs between PDGs and SRGs where verification can be done?

Main references

- [1] **D.E. Muller and P.E. Schupp**: The theory of ends, pushdown automata, and second-order logic, TCS 37(1), 1985.
- [2] **C. Morvan and C. Stirling**: Rational graphs trace context-sensitive languages, MFCS01.
- [3] **C. Rispal**: The synchronized graphs trace the context-sensitive languages, ENTCS 68(6), 2002.
- [4] **M. Latteux and D. Simplot**: Context-sensitive string languages and recognizable picture languages, INF COMPUT 138(2), 1997.
- [5] **W. Thomas**: A short introduction to infinite automata, DLT01.