Finite Automata Over Infinite Alphabets: Two Models with Transitions for Local Change

Christopher Czyba   Christopher Spinrath   Wolfgang Thomas

RWTH Aachen University

DLT 2015

Background

Recognizing languages over infinite alphabets

- Example: alphabet of the natural numbers $\mathbb{N}$
- Verification (e.g. counters)
- Database Theory ("data words")

- Register Automata, *Francez, Kaminski*
- Pebble Automata, *Milo, Neven, Schwentick, Suciu, Vianu*

$\rightsquigarrow$ (non-)equality of input letters

Background

Recognizing languages over infinite alphabets

- Example: alphabet of the natural numbers $\mathbb{N}$

- Verification (e.g. counters)
- Database Theory ("data words")

- Register Automata, *Francez, Kaminski*
- Pebble Automata, *Milo, Neven, Schwentick, Suciu, Vianu*

$\rightsquigarrow$ (non-)equality of input letters

Here: local change rather than (non-)equality of input letters

Example: $\{0\ 1\ 2\ \cdots\ n \mid n \in \mathbb{N}\} \subset \mathbb{N}^*$

Outline

Strong Automata – Definition

- Introduced by Spelten, Thomas, Winter
- Idea: "compare" successive letters via logical formulae

Strong Automata – Definition

- Introduced by Spelten, Thomas, Winter
- Idea: “compare” successive letters via logical formulae

Strong Automata are “finite automata”

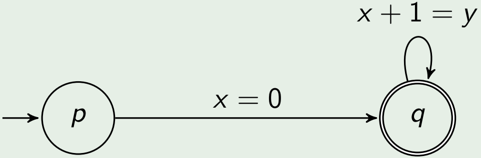
- the alphabet is $\mathbb{N}$
- the transition format is $p,\ \varphi(x,y),\ q$

  Move from p to q via letter n with previous letter m, if $\varphi[m,n]$ is true

- the model depends on a logic

Strong Automata – Example

Example



- Recognized language: $\{0\ 1\ 2\ \cdots\ n \mid n \in \mathbb{N}\}$

Strong Automata – Closure Properties

Lemma

Given a strong automaton $\mathfrak{A}$, one can construct a deterministic strong automaton $\mathfrak{A}'$ such that $L(\mathfrak{A}) = L(\mathfrak{A}')$.

Proof idea:

- adaption of the classical powerset construction using Boolean combinations of the given transition formulae

Strong Automata – Closure Properties

Lemma

Given a strong automaton $\mathfrak{A}$, one can construct a deterministic strong automaton $\mathfrak{A}'$ such that $L(\mathfrak{A}) = L(\mathfrak{A}')$.

Proof idea:

- adaption of the classical powerset construction using Boolean combinations of the given transition formulae

Proposition

The languages recognized by strong automata form an effective Boolean algebra.

Strong Automata – The Emptiness Problem

Theorem

The non-emptiness problem for strong automata with transition formulae in MSO logic over $(\mathbb{N}, +1)$ is decidable.

Proof: using Büchi's decidability result on the MSO-theory of $(\mathbb{N}, +1)$ (description of transitive closure in MSO logic).

Strong Automata – Extensions

Extensions

1. Connect more than two successive letters
2. Lift the input alphabet to $\mathbb{N} \times \mathbb{N}$

Strong Automata – Extensions

Extensions

1. Connect more than two successive letters
2. Lift the input alphabet to $\mathbb{N} \times \mathbb{N}$

In both cases:

- Transition formulae have more than two free variables
- The emptiness problem is undecidable for $(\mathbb{N}, +1)$ and FO logic

Proof idea:

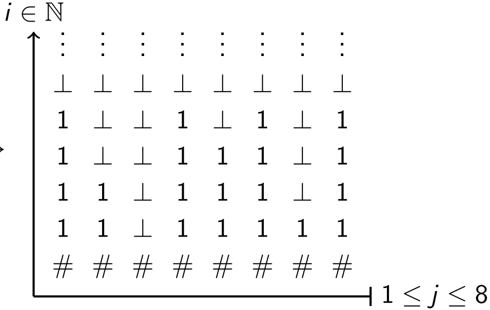
- Reduce the reachabillity problem for 2-register machines

Outline

Progressive Grid Automata – Grid Words

A word over $\mathbb{N}$ is viewed as a grid word:

$4\ 2\ 0\ 4\ 3\ 4\ 1\ 4 \mapsto$



Progressive Grid Automata – Grid Words

A word over $\mathbb{N}$ is viewed as a grid word:

4 2 0 4 3 4 1 4 $\mapsto$

$i \in \mathbb{N}$							
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
1	$\perp$	$\perp$	1	$\perp$	1	$\perp$	1
1	$\perp$	$\perp$	1	1	1	$\perp$	1
1	1	$\perp$	1	1	1	$\perp$	1
1	1	$\perp$	1	1	1	1	1
#	#	#	#	#	#	#	#

$1 \leq j \leq 8$

- The bottom row is labeled #
- In each column from some point on $\perp$
- Remaining positions are labeled with 1

Progressive Grid Word Automata – Definition

Definition

A Progressive Grid Automaton is a tuple $\mathcal{A} = (Q, \delta, q_0, F)$ where

- Q is a finite set of states,
- Δ is the transition relation with

$$\Delta \subseteq Q \times \{\#, \perp, 1\} \times Q \times \{\uparrow, \downarrow, \rightarrow\}$$

- $q_0 \in Q$ is the initial state and
- $F \subseteq Q$ is the set of accepting states.

Progressive Grid Word Automata – Definition

Definition

A Progressive Grid Automaton is a tuple $\mathcal{A} = (Q, \delta, q_0, F)$ where

- Q is a finite set of states,
- Δ is the transition relation with

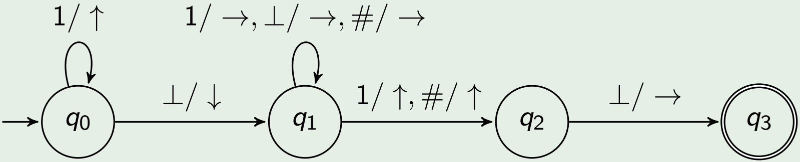
$$\Delta \subseteq Q \times \{\#, \perp, 1\} \times Q \times \{\uparrow, \downarrow, \rightarrow\}$$

$$\Delta \cap Q \times \{\#\} \times Q \times \{\downarrow\} = \emptyset,$$

- $q_0 \in Q$ is the initial state and
- $F \subseteq Q$ is the set of accepting states.

Progressive Grid Automata – Example

Example



$$L := \{n_0 \dots n_p \in \mathbb{N}^* \mid p > 0 \wedge n_0 = n_p\}$$

Progressive Grid Automata – Closure Properties

	deterministically	non-deterministically
Complement	Yes ✓	No
Intersection	No	No
Union	No	Yes ✓

- Complement: method to ensure termination of a computation in a column
- Intersection: no details here (too technical)

Progressive Grid Automata – Closure Properties

	deterministically	non-deterministically
Complement	Yes ✓	No
Intersection	No	No
Union	No	Yes ✓

- Complement: method to ensure termination of a computation in a column
- Intersection: no details here (too technical)

Corollary

Non-determinism is strictly more expressive than determinism.

Outline

Progressive Grid Automata – The Emptiness Problem

Theorem

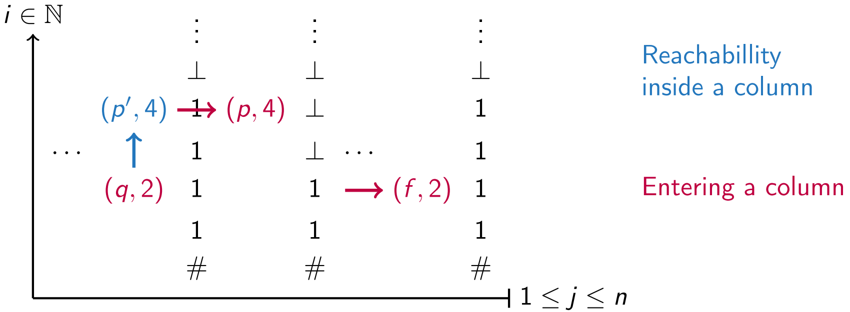
The emptiness problem for progressive grid automata is decidable.

Progressive Grid Automata – The Emptiness Problem

Theorem

The emptiness problem for progressive grid automata is decidable.

Proof approach: consider only the vertical position:



Progressive Grid Automata – The Emptiness Problem

Entering a column

- Initially, any column can be entered in configuration $(q_0, 1)$.

Progressive Grid Automata – The Emptiness Problem

Entering a column

- Initially, any column can be entered in configuration $(q_0, 1)$.
- If there is a column that can be entered in (q, k),
    - some (p', ℓ) can be reached inside this column
    - and there is a suitable horizontal transition to p
    - then any column may be entered in (p, ℓ).

Progressive Grid Automata – The Emptiness Problem

Entering a column

- Initially, any column can be entered in configuration $(q_0, 1)$.
- If there is a column that can be entered in (q, k),
  - some (p', ℓ) can be reached inside this column
  - and there is a suitable horizontal transition to p
  - then any column may be entered in (p, ℓ).

$\rightsquigarrow$ least fixed point definable in MSO logic over $Q \times (\mathbb{N}, +1)$
$\rightsquigarrow$ again, apply Büchi's decidability result (also true here)

Progressive Grid Automata – The Emptiness Problem

Entering a column

- Initially, any column can be entered in configuration $(q_0, 1)$.
- If there is a column that can be entered in (q, k),
  - some (p', ℓ) can be reached inside this column
  - and there is a suitable horizontal transition to p
  - then any column may be entered in (p, ℓ).

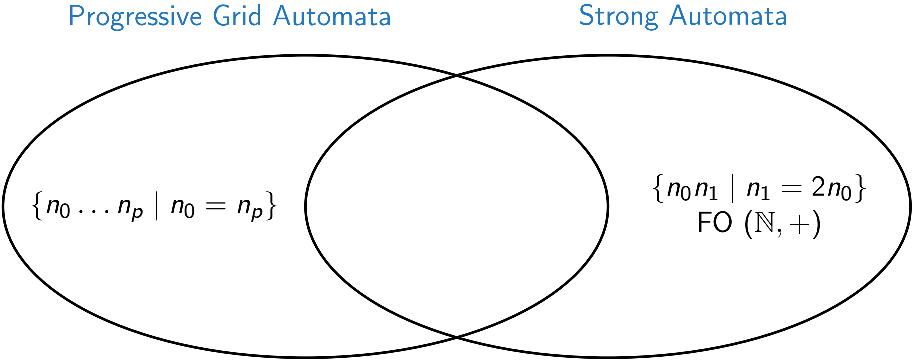
$\rightsquigarrow$ least fixed point definable in MSO logic over $Q \times (\mathbb{N}, +1)$
$\rightsquigarrow$ again, apply Büchi's decidability result (also true here)

Reachabillity inside a column

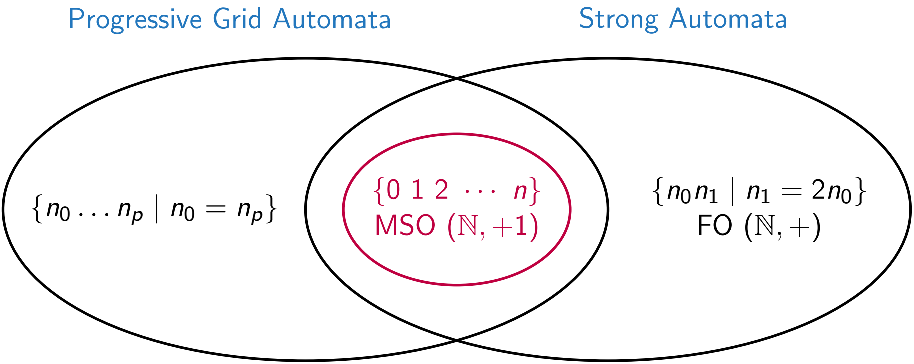
- Can be defined analogously (by a least fixed point)

Outline

Comparison – Languages over $\mathbb{N}$



Comparison – Languages over $\mathbb{N}$



Conclusion

Summary

Two models with transitions for local change:

- Strong Automata
- Progressive Grid Automata
- The Emptiness Problem is decidable
- Different expressive power

Further Prospects

- Extension to other infinite alphabets like Σ^* rather than $\mathbb{N}$
- Extension to infinite words
- General framework of progressive grid automata
- Complexity of the decision problems