Lecture Verification of Infinite-State Systems

Winter 2026/2027

Prof. Dr. Karin Quaas

Institut für Informatik
Universität Leipzig

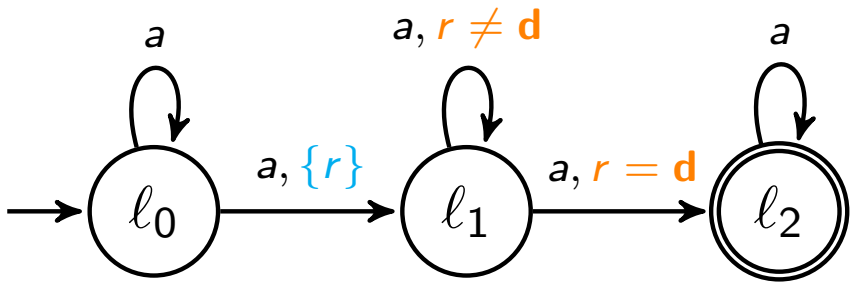
Lecture 2 - Register Automata

Schedule

	Thursday 9-11	Friday 13-15
12.10.-18.10.		Lecture 1: Introduction / Register Automata
19.10.-25.10.	Exercise 1	Lecture 2: Register Automata
26.10.-1.11.	Exercise 2	Lecture 3: Register Automata
2.11.-8.11.	Exercise 3	Lecture 4: Register Automata
9.11.-15.11.	Exercise 4	Lecture 5: Register Automata / Two-Counter Mac
16.11.-22.11.	Exercise 5	Lecture 6: Register Automata
23.11.-29.11.	Exercise 6	Lecture 7: Freeze LTL
30.11.-6.12.	Exercise 7	Lecture 8: Freeze LTL
7.12.-13.12.	Exercise 8	Lecture 9: Freeze LTL / Alternating RA
14.12.-20.12.	Exercise 9	Lecture 10: Alternating RA / WSTS
4.1.-10.1.	Exercise 10	Lecture 11: Alternating RA / WSTS
11.1.-17.1.	Exercise 11	Lecture 12: Alternating RA / WSTS
18.1.-24.1.	Exercise 12	Lecture 13: Alternating RA / Ackermann
25.1.-31.1.	Exercise 13	Lecture 14:
1.2.-7.2.	Exercise 14	Examination

Recap Register Automata

$\mathcal{A}$



The following

$$(\ell_0, \bot) \xrightarrow{(a,2)} (\ell_0, \bot) \xrightarrow{(a,3)} (\ell_1, 3) \xrightarrow{(a,8)} (\ell_1, 3) \xrightarrow{(a,4)} (\ell_1, 3) \xrightarrow{(a,3)} (\ell_2, 3)$$

is a run on the data word $(a,2)(a,3)(a,8)(a,4)(a,3)$

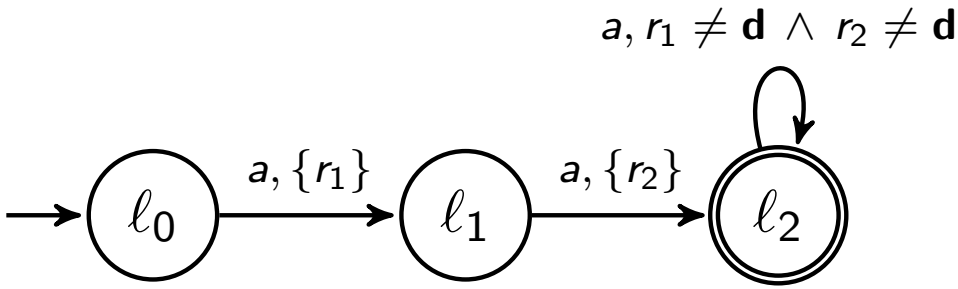
$L(\mathcal{A}) = \{(a, d_1)(a, d_2)\ldots(a_n, d_n) \mid$ there exist $1 \leq i < j \leq n$
such that $d_i = d_j$ and $d_k \neq d_i$ for all $i < k < j, n \geq 2\}$

Warm Up

Let $L \subseteq (\{a\} \times \mathbb{N})^*$ be the data language that consists of all data words w such that w contains at least two letters, and the first two data values never appear again in w.

Is L data regular?

L is data regular because it is accepted by the following register automaton.



This register automaton is deterministic... wait, what does deterministic actually mean?

Syntactic vs. Semantic Definition of Determinism

Let $\mathcal{A} = (Q, \Sigma, Q_{\text{in}}, \Delta, Q_{\text{acc}})$ be a traditional finite automaton.

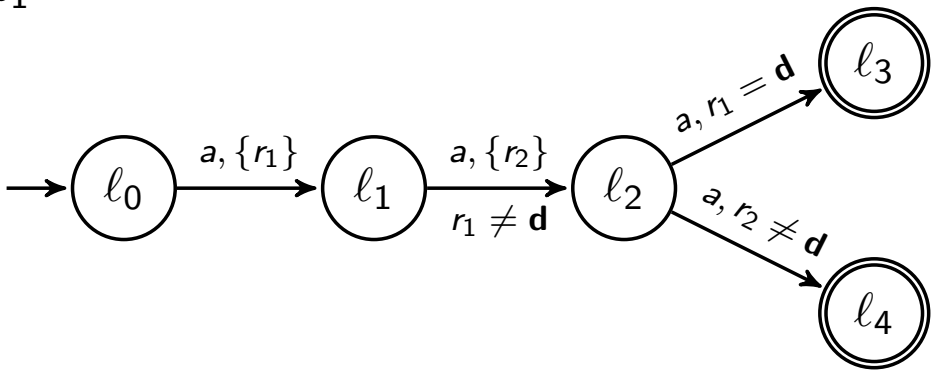
We say that $\mathcal{A}$ is deterministic if for all $q \in Q$ and $a \in \Sigma$, there exists at most one $q' \in Q$ such that $(q, a, q') \in \Delta$.

- This is a purely syntactic definition of determinism
- It can be decided in **P** whether a given finite automaton is deterministic
- If $\mathcal{A}$ is deterministic, then there exists at most one initialized run of $\mathcal{A}$ on w, for every input word $w \in \Sigma^*$ (semantic definition of determinism)
- Note that the converse holds as well if $\mathcal{A}$ does not contain any nonreachable states (which we assume wlog)

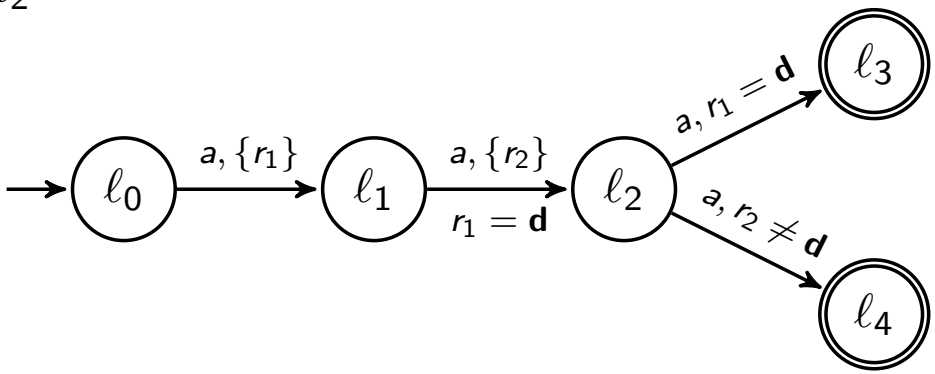
Determinism for Register Automata

Are the register automata $\mathcal{A}_1$ and $\mathcal{A}_2$ depicted below deterministic?

$\mathcal{A}_1$



$\mathcal{A}_2$

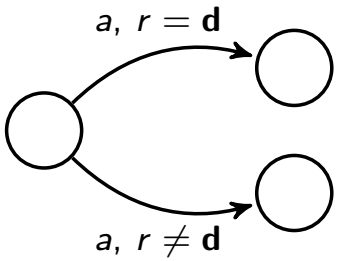


Deterministic Register Automata

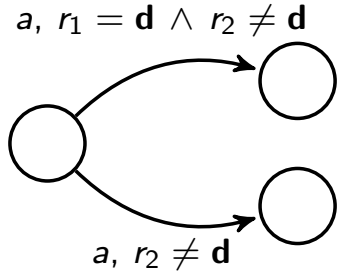
A register automaton $\mathcal{A} = (\mathcal{L}, \Sigma, r, \mathcal{L}_{\text{in}}, E, \mathcal{L}_{\text{acc}})$ is deterministic if

- $\mathcal{L}_{\text{in}}$ is a singleton ($|\mathcal{L}_{\text{in}}| = 1$), and
- for all edges $(\ell, a, \phi_1, \lambda_1, \ell_1) \neq (\ell, a, \phi_2, \lambda_2, \ell_2)$, the register test $\phi_1 \wedge \phi_2$ is not satisfiable.

Deterministic:



Not Deterministic:



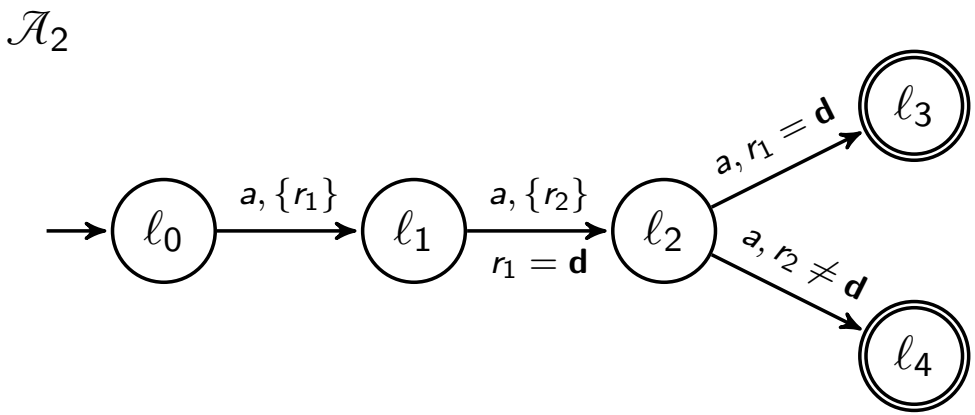
Deterministic Register Automata

A register automaton $\mathcal{A} = (\mathcal{L}, \Sigma, r, \mathcal{L}_{\text{in}}, E, \mathcal{L}_{\text{acc}})$ is deterministic if

- $\mathcal{L}_{\text{in}}$ is a singleton ($|\mathcal{L}_{\text{in}}| = 1$), and
- for all edges $(\ell, a, \phi_1, \lambda_1, \ell_1) \neq (\ell, a, \phi_2, \lambda_2, \ell_2)$, the register test $\phi_1 \wedge \phi_2$ is not satisfiable.

- It can be decided in **P** whether a given register automaton $\mathcal{A}$ is deterministic: checking whether $\phi_1 \wedge \phi_2$ is not satisfiable can be done in **P** (see Exercise 1.4)
- If $\mathcal{A}$ is deterministic, then for every input data word $w \in (\Sigma \times \mathbb{N})^*$ there exists at most one initialized run of $\mathcal{A}$ on w
- The converse does not hold

Deterministic Register Automata



$\mathcal{A}_2$ is not deterministic : $r_1 = \mathbf{d} \wedge r_2 \neq \mathbf{d}$ is satisfiable.

But for every input data word $w \in (\Sigma \times \mathbb{N})^*$ there there exists at most one initialized run.

Deterministic Register Automata

A register automaton $\mathcal{A} = (\mathcal{L}, \Sigma, r, \mathcal{L}_{\text{in}}, E, \mathcal{L}_{\text{acc}})$ is deterministic if

- $\mathcal{L}_{\text{in}}$ is a singleton ($|\mathcal{L}_{\text{in}}| = 1$), and
- for all edges $(\ell, a, \phi_1, \lambda_1, \ell_1) \neq (\ell, a, \phi_2, \lambda_2, \ell_2)$, the register test $\phi_1 \wedge \phi_2$ is not satisfiable.

- It can be decided in **P** whether a given register automaton $\mathcal{A}$ is deterministic: checking whether $\phi_1 \wedge \phi_2$ is not satisfiable can be done in **P** (see Exercise 1.4)
- If $\mathcal{A}$ is deterministic, then for every input data word $w \in (\Sigma \times \mathbb{N})^*$ there exists at most one initialized run of $\mathcal{A}$ on w.
- The converse does not hold

Syntactic vs. Semantic Definition of Completeness

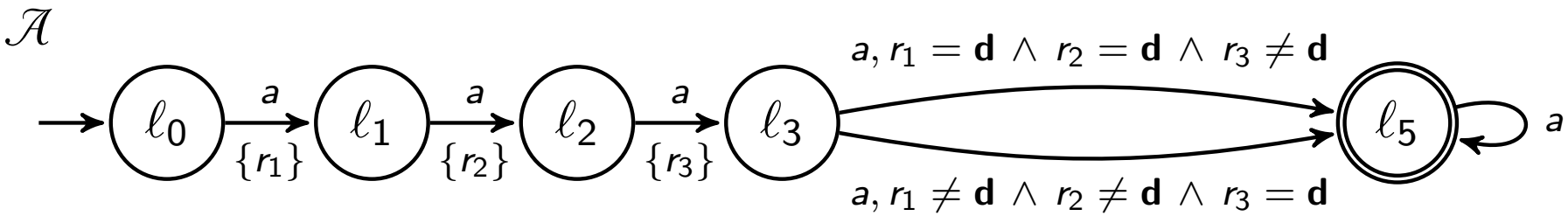
Let be a $\mathcal{A} = (Q, \Sigma, Q_{\text{in}}, \Delta, Q_{\text{acc}})$ be a traditional finite automaton

We say that $\mathcal{A}$ is complete if for all $q \in Q$ and $a \in \Sigma$, there exists at least one $q' \in Q$ such that $(q, a, q') \in \Delta$.

- This is is a purely syntactic definition of completeness
- It can be checked in **P** whether a given finite automaton is complete
- If $\mathcal{A}$ is complete, then there exists at least one initialized run of $\mathcal{A}$ on w, for every input word $w \in \Sigma^*$ (semantic definition of completeness)
- Note that the converse holds as well if $\mathcal{A}$ does not contain any nonreachable states and $\mathcal{A}$ is deterministic

Completeness for Register Automata

Is the register automaton $\mathcal{A}$ depicted below complete?



$\mathcal{A}$ is not complete:

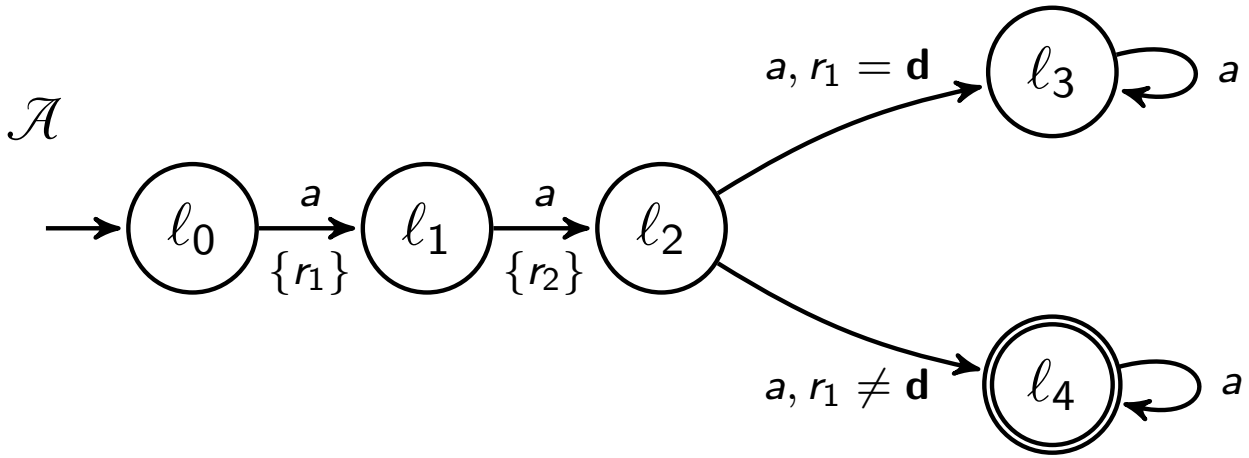
- for $(\ell_3, (1,1,1))$ and $d = 1$ there does not exist any edge $(\ell_3, a, \phi, \lambda, \ell')$ such that $\nu, d \models \phi$
- Hence there does not exist any initialized run of $\mathcal{A}$ on $(a,1)(a,1)(a,1)(a,1)$

All Register Tests

- How many different cases for $\nu, d \models \phi$ are relevant?
- If $\mathcal{R} = \{r_1, \ldots, r_n\}$, there are 2^n different relevant cases
- Each of these cases should be covered
- For example, for $n = 2$, we need
  - $r_1 = \mathbf{d} \wedge r_2 = \mathbf{d}$
  - $r_1 = \mathbf{d} \wedge r_2 \neq \mathbf{d}$
  - $r_1 \neq \mathbf{d} \wedge r_2 = \mathbf{d}$
  - $r_1 \neq \mathbf{d} \wedge r_2 \neq \mathbf{d}$
- Use $\Phi^{\mathsf{cpl}}(\mathcal{R})$ to denote this set of 2^n register tests

Completeness for Register Automata

Is the register automaton $\mathcal{A}$ depicted below complete?



$\mathcal{A}$ is complete: for every input data word $w \in (\Sigma \times \mathbb{N})^*$ there exists some initialized run of $\mathcal{A}$ on w

Note that $r_1 = \mathbf{d}$ and $r_1 \neq \mathbf{d}$ are not in $\Phi^{\text{cpl}}(\mathcal{R})$, but for every register test $\phi \in \Phi^{\text{cpl}}(\mathcal{R})$ we have ϕ entails $r_1 = \mathbf{d}$ or $r_1 \neq \mathbf{d}$

Register Test Entailment

Given register tests ϕ and ϕ', we say that ϕ entails ϕ', written $\phi \models \phi'$, if

$$\nu, d \models \phi \quad \Longrightarrow \quad \nu, d \models \phi'$$

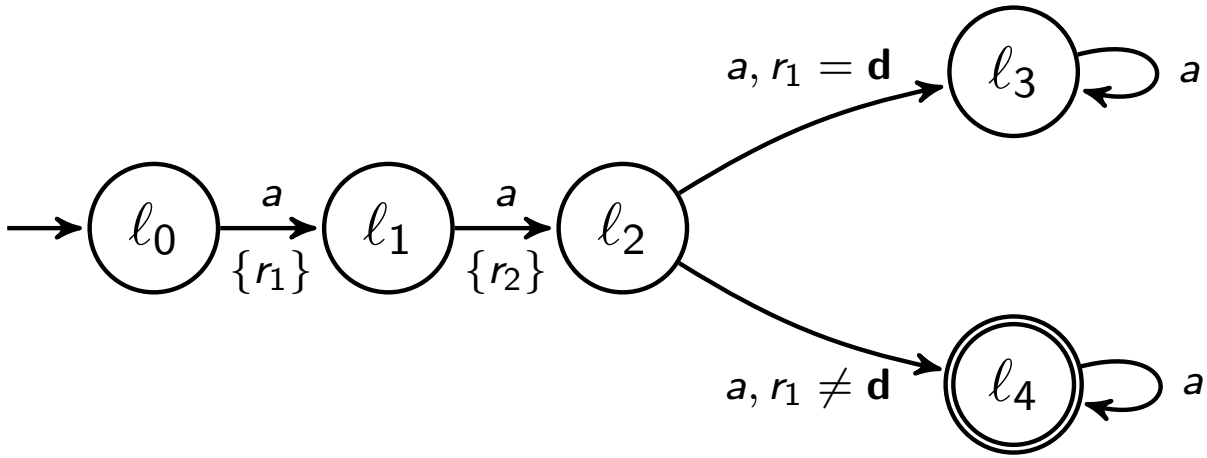
for every register valuation ν and datum $d \in \mathbb{N}$.

Examples

- $\phi = (r_1 \neq \mathbf{d})$ and $\phi' = (r_1 \neq \mathbf{d})$: $\phi \models \phi'$
- $\phi = (r_2 \neq \mathbf{d})$ and $\phi' = (r_1 \neq \mathbf{d})$: $\phi \not\models \phi'$
- $\phi = (r_1 \neq \mathbf{d})$ and $\phi' = \texttt{true}$: $\phi \models \phi'$
- $\phi = (r_1 = \mathbf{d} \wedge r_2 = \mathbf{d})$ and $\phi' = (r_1 = \mathbf{d})$: $\phi \models \phi'$
- $\phi = \texttt{true}$ and $\phi' = (r_1 = \mathbf{d})$: $\phi \not\models \phi'$

Complete Register Automata

A register automaton $\mathcal{A} = (\mathcal{L}, \Sigma, \mathcal{R}, \mathcal{L}_{\text{in}}, E, \mathcal{L}_{\text{acc}})$ is complete
if for all $\ell \in \mathcal{L}$, $a \in \Sigma$, and $\phi \in \Phi^{\text{cpl}}(\mathcal{R})$
there exists some edge $(\ell, a, \phi', \lambda, \ell') \in E$ such that $\phi \models \phi'$.



Complete Register Automata

Let us prove that if $\mathcal{A}$ is complete, then for every input data word $w = (a_1, d_1) \dots (a_n, d_n)$ there exists at least one initialized run of $\mathcal{A}$ on w.

The proof is by induction on n.For the induction base, set $n = 0$. Then (ℓ_0, ν_0) with $\ell_0 \in \mathcal{L}_{\text{in}}$ and $\nu_0 = \nu_\perp$ is an initialized run of $\mathcal{A}$ on ε.

Suppose the claim holds for all data words of length $n-1$ for some $n \geq 1$. We prove it holds for data words of length n as well.

By induction hypothesis, there exists an initialized run

$$(\ell_0, \nu_0) \xrightarrow{(a_1,d_1)} \dots \xrightarrow{(a_{n-1},d_{n-1}} (\ell_{n-1}, \nu_{n-1})$$

of $\mathcal{A}$ on $(a_1, d_1) \dots (a_{n-1}, d_{n-1})$. Let $\phi \in \Phi^{\text{cpl}}(\mathcal{R})$ be the unique register test such that $\nu_{n-1}, d_n \models \phi$. By assumption, there exists edge $(\ell_{n-1}, a_n, \phi_n, \lambda_n, \ell_n)$ such that $\phi \models \phi_n$. Hence $\nu_{n-1}, d_n \models \phi_n$. Let $\nu_n = \nu_{n-1}[\lambda_n \leftarrow d_n]$. Then $(\ell_{n-1}, \nu_{n-1}) \xrightarrow{(a_n,d_n)} (\ell_n, \nu_n)$, and $(\ell_0, \nu_0) \xrightarrow{(a_1,d_1)} \dots \xrightarrow{(a_{n-1},d_{n-1})} (\ell_{n-1}, \nu_{n-1}) \xrightarrow{(a_n,d_n)} (\ell_n, \nu_n)$ is an initialized run of $\mathcal{A}$ on w.

Completing Deterministic Register Automata

Theorem 2

For every deterministic register automaton $\mathcal{A}$ there exists a complete deterministic register automaton $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.

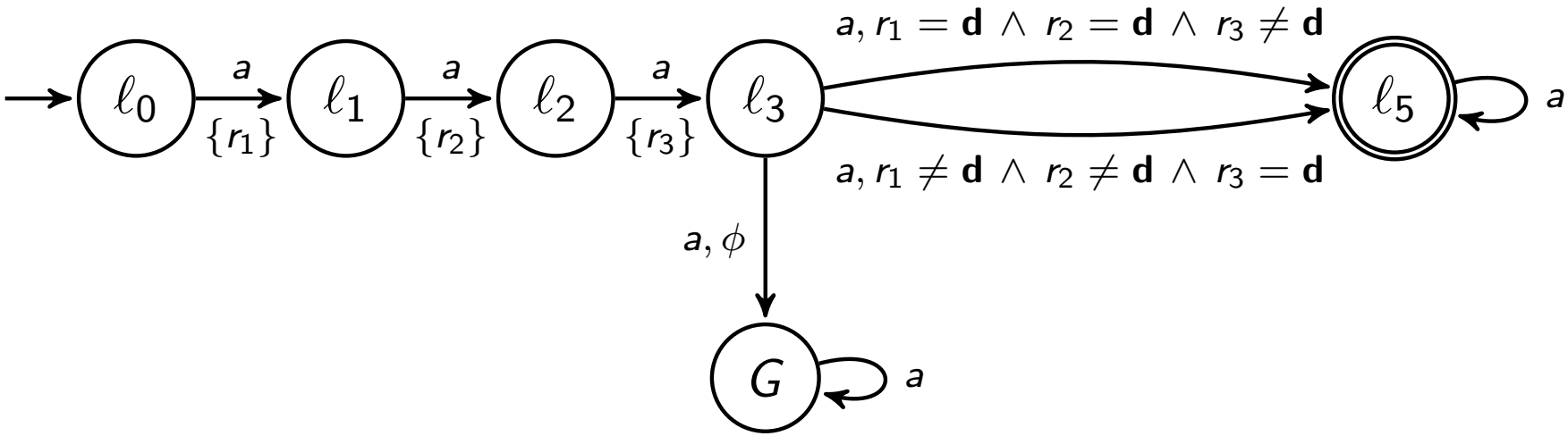
Proof.

Let $\mathcal{A} = (\mathcal{L}, \Sigma, \mathcal{R}, \mathcal{L}_{\text{in}}, E, \mathcal{L}_{\text{acc}})$ be a deterministic register automaton. For obtaining a complete register automaton, we run the following algorithm on $\mathcal{A}$:

- add a new "garbage location" G with selfloop $(G, a, \texttt{true}, \emptyset, G)$ for all $a \in \Sigma$
- for every $\ell \in \mathcal{L}$, $a \in \Sigma$, and $\phi \in \Phi^{\text{cpl}}(\mathcal{R})$, check whether there exists some edge $(\ell, a, \phi', \lambda, \ell') \in E$ such that $\phi \models \phi'$.
- If yes, do nothing
- If no, add edge $(\ell, a, \phi, \emptyset, G)$



Complete Deterministic Register Automata Example



ϕ is the disjunction of the following register tests:

- $r_1 = \mathbf{d} \wedge r_2 = \mathbf{d} \wedge r_3 = \mathbf{d}$
- $r_1 = \mathbf{d} \wedge r_2 \neq \mathbf{d} \wedge r_3 = \mathbf{d}$ and $r_1 = \mathbf{d} \wedge r_2 \neq \mathbf{d} \wedge r_3 \neq \mathbf{d}$
- $r_1 \neq \mathbf{d} \wedge r_2 \neq \mathbf{d} \wedge r_3 \neq \mathbf{d}$
- $r_1 \neq \mathbf{d} \wedge r_2 = \mathbf{d} \wedge r_3 = \mathbf{d}$ and $r_1 \neq \mathbf{d} \wedge r_2 = \mathbf{d} \wedge r_3 \neq \mathbf{d}$

Are These Data Languages Data Regular?

Let $\Sigma = \{a, b\}$

- $L_1 = \{(a, 5)(b, 7)\}$
- $L_2 = \{(a, d)(b, d') \mid d \neq d'\}$
- $L_3 = \Sigma \times \mathbb{N}$
- $L_4 = \{\varepsilon\}$
- $L_5 = \{(a_1, d_1) \dots (a_n, d_n) \mid$ there exist $1 \leq i < j \leq n$ such that $d_i = d_j, n \geq 2\}$
- $L_6 = \{(a_1, d_1) \dots (a_n, d_n) \mid d_i = d_j$ for all $1 \leq i < j \leq n, n \geq 1\}$
- $L_7 = \{(a_1, d_1) \dots (a_n, d_n) \mid d_i \neq d_j$ for all $1 \leq i < j \leq n, n \geq 0\}$
- $L_8 = \{(a_1, d_1) \dots (a_n, d_n) \mid d_i \neq d_{i+1}$ for all $1 \leq i < n, n \geq 0\}$
- $L_9 = \{(a_1, d_1) \dots (a_n, d_n) \mid d_1 < d_2 < \cdots < d_n, n \geq 0\}$
- $L_{10} = \{(a_1, d_1) \dots (a_n, d_n) \mid a_1 a_2 \dots a_n$ is a palindrome, $n \geq 0\}$

L_1 is not data regular

We prove that $L_1 = \{(a,5)(b,7)\}$ is not data regular. Towards contradiction, suppose there exists some register automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L_1$. Then there exists an initialized accepting run of $\mathcal{A}$ on $(a,5)(b,7)$ of the form

$$(\ell_0, \nu_0) \xrightarrow{(a,5)} (\ell_1, \nu_1) \xrightarrow{(b,7)} (\ell_2, \nu).$$

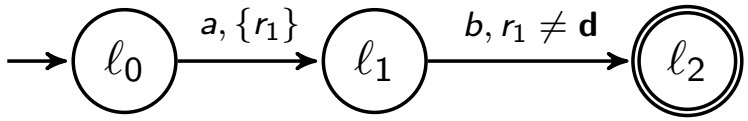
By definition, there must exist some edge $(\ell_1, b, \phi, \lambda, \ell_2)$ in $\mathcal{A}$ such that $\nu_1, 7 \models \phi$. Note that $\nu_1(r) \in \{\bot, 5\}$ for all registers r in $\mathcal{A}$. Hence, the register test ϕ can only use inequality tests of the form $r \neq \mathbf{d}$ or `true`, as otherwise $\nu_1, 7 \models \phi$ cannot hold. But then we also have $\nu_1, 8 \models \phi$, so that

$$(\ell_0, \nu_0) \xrightarrow{(a,5)} (\ell_1, \nu_1) \xrightarrow{(b,8)} (\ell_2, \nu)$$

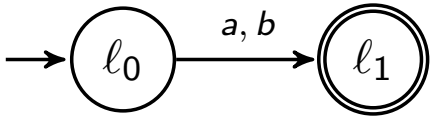
is an initialized accepting run of $\mathcal{A}$ on $(a,5)(b,8)$ too. Contradiction.

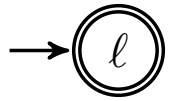
L_2, L_3 and L_4 are data regular

The data language $L_2 = \{(a, d)(b, d') \mid d \neq d'\}$ can be accepted by the following register automaton:



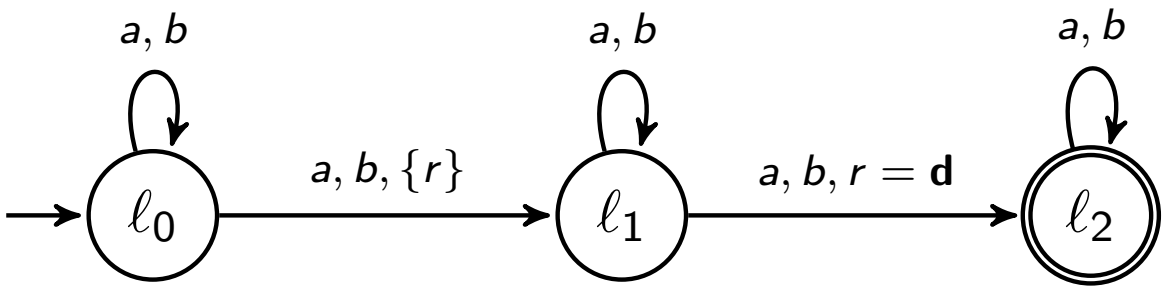
The data languages $L_3 = \Sigma \times \mathbb{N}$ and $L_4 = \{\varepsilon\}$ can be accepted, too:





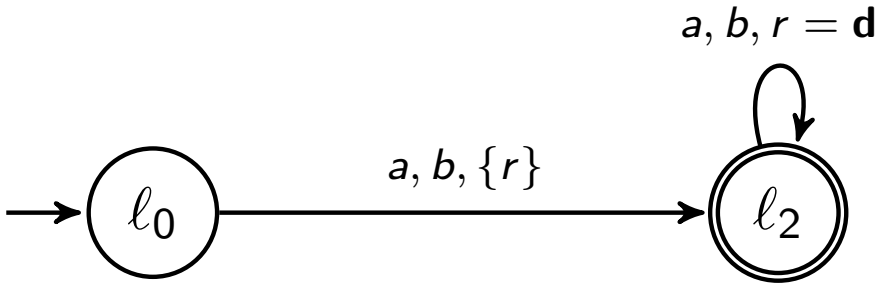
L_5 is data regular

The following register automaton accepts $L_5 = \{(a_1, d_1) \dots (a_n, d_n) \mid$ there exist $1 \leq i < j \leq n$ such that $d_i = d_j, n \geq 2\}$



L_6 is data regular

The following register automaton accepts
$L_6 = \{(a_1, d_1) \dots (a_n, d_n) \mid d_i = d_j \text{ for all } 1 \leq i < j \leq n, n \geq 1\}$



We can (easier) write L_6 as
$\{(a_1, d_1) \dots (a_n, d_n) \mid d_i = d_1 \text{ for all } 1 \leq i \leq n, n \geq 1\}$

Note that the register automaton above is deterministic.

L_7 is not data regular

$L_7 = \{(a_1, d_1) \dots (a_n, d_n) \mid d_i \neq d_j \text{ for all } 1 \leq i < j \leq n, n \geq 1\}$ cannot be accepted by any register automaton. Towards contradiction suppose there exists some register automaton $\mathcal{A}$ with $L(\mathcal{A}) = L_7$. Suppose $\mathcal{A}$ uses k registers. Consider the data word

$$(a, d_1)(a, d_2) \dots (a, d_k)(a, d_{k+1})(a, d_{k+2}),$$

where $d_i \neq d_j$ for all $1 \leq i < j \leq k+2$. Hence $w \in L_7$ and thus, there must exist some initialized accepting run of $\mathcal{A}$ on w of the form

$$(\ell_0, \nu_0) \xrightarrow{(a,d_1)} \dots \xrightarrow{(a,d_k)} (\ell_k, \nu_k) \xrightarrow{(a,d_{k+1})} (\ell_{k+1}, \nu_{k+1}) \xrightarrow{(a,d_{k+2})} (\ell_{k+2}, \nu_{k+2})$$

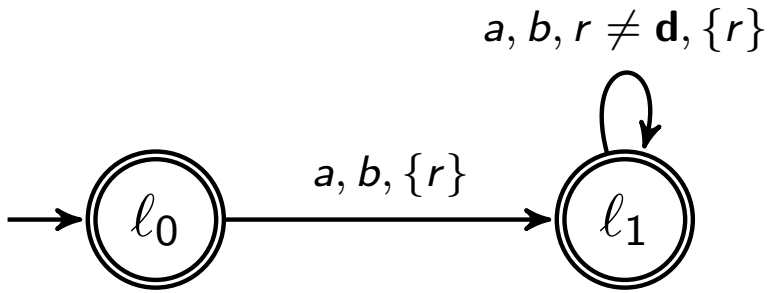
There exists some $1 \leq i \leq k+1$ such that $d_i \notin \nu_{k+1}(r)$ for all registers r in $\mathcal{A}$. Hence

$$(\ell_0, \nu_0) \xrightarrow{(a,d_1)} \dots \xrightarrow{(a,d_k)} (\ell_k, \nu_k) \xrightarrow{(a,d_{k+1})} (\ell_{k+1}, \nu_{k+1}) \xrightarrow{(a,d_i)} (\ell_{k+2}, \nu_{k+2})$$

is an initialized accepting run of $\mathcal{A}$ on some data word not in L_7, contradiction.

L_8 is data regular

The following register automaton accepts
$L_8 = \{(a_1, d_1) \dots (a_n, d_n) \mid d_i \neq d_{i+1} \text{ for all } 1 \leq i < n, n \geq 0\}$



L_9 is not data regular

The data language $L_9 = \{(a_1, d_1) \dots (a_n, d_n) \mid d_1 < d_2 < \cdots < d_n, n \geq 0\}$ is not data regular.
Towards contradiction, suppose there exists some register automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L_9$. Then there exists an initialized accepting run of $\mathcal{A}$ on $(a, 5)(b, 7)$ of the form

$$(\ell_0, \nu_0) \xrightarrow{(a,5)} (\ell_1, \nu_1) \xrightarrow{(b,7)} (\ell_2, \nu).$$

By definition, there must exist some edge $(\ell_1, b, \phi, \lambda, \ell_2)$ in $\mathcal{A}$ such that $\nu_1, 7 \models \phi$. Note that $\nu_1(r) \in \{\bot, 5\}$ for all registers r in $\mathcal{A}$. Hence, the register test ϕ can only use inequality tests of the form $r \neq \mathbf{d}$ or `true`, as otherwise $\nu_1, 7 \models \phi$ cannot hold. But then we also have $\nu_1, 3 \models \phi$, so that

$$(\ell_0, \nu_0) \xrightarrow{(a,5)} (\ell_1, \nu_1) \xrightarrow{(b,3)} (\ell_2, \nu)$$

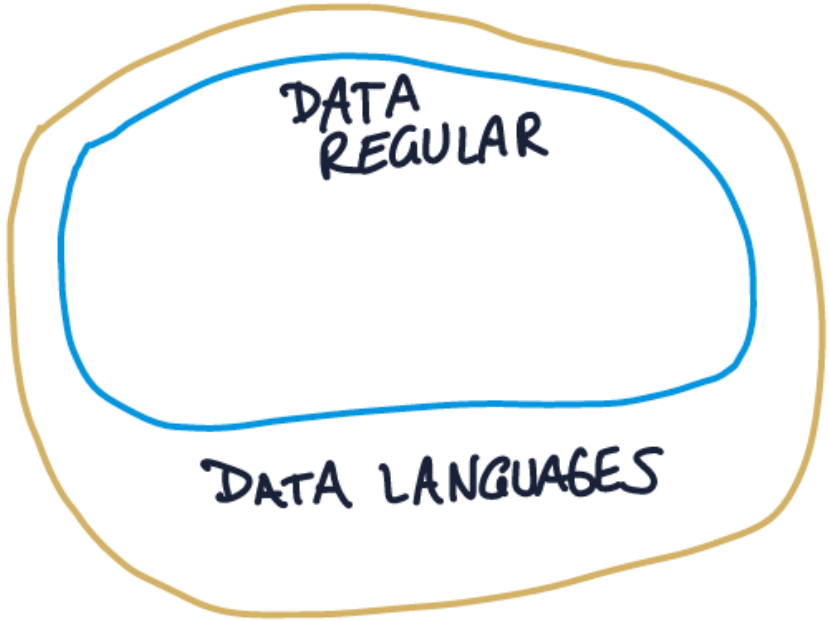
is an initialized accepting run of $\mathcal{A}$ on $(a, 5)(b, 3)$ too. Contradiction.

L_{10}

The data language
$L_{10} = \{(a_1, d_1) \dots (a_n, d_n) \mid a_1 a_2 \dots a_n \text{ is a palindrome}, n \geq 0\}$ is...

... will come soon.

Data Languages over $(\mathbb{N}; =)$



- There are data languages that are not data regular
- Possible remedies:
  - Relational structures other than $(\mathbb{N}; =)$
  - Generalisations of register automata, e.g., alternating register automata, register context-free grammars