

Formalisms for Data Languages

Part I: Register Automata and Freeze LTL

Karin Quaas

Universität Leipzig

QuantLA SpringSchool 2016

Data Languages

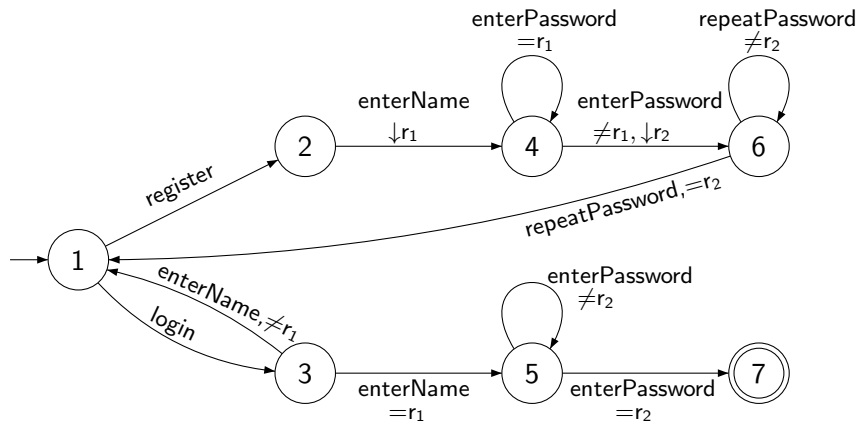
Σ ...a finite alphabet

D ...an infinite data domain, e.g., \mathbb{N} , $\mathbb{R}_{\geq 0}$, ASCII strings

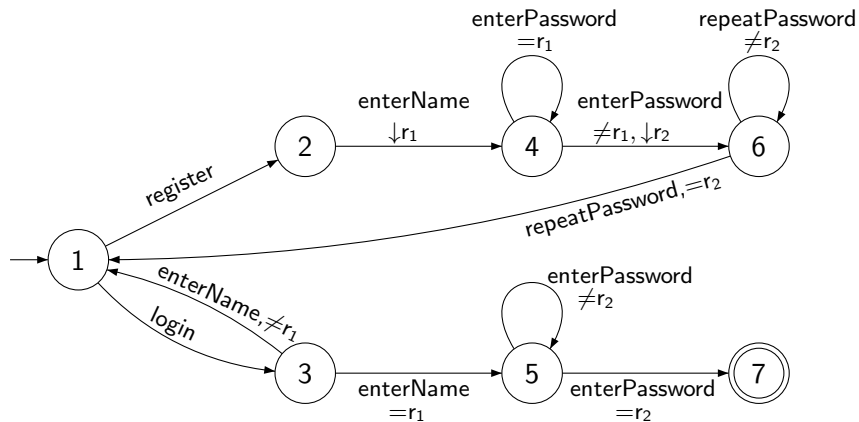
A **data word** is a finite sequence $(a_1, d_1)(a_2, d_2) \dots (a_n, d_n)$ in $(\Sigma \times D)^*$.

A **data language** is a subset of $(\Sigma \times D)^*$.

Register Automata

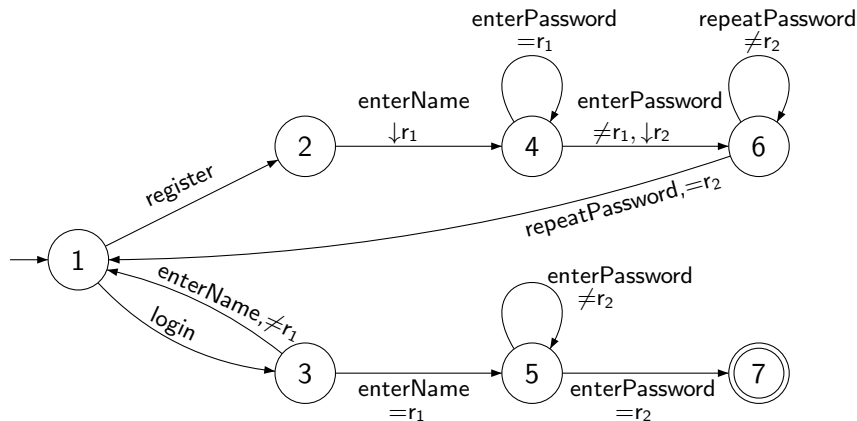


Register Automata



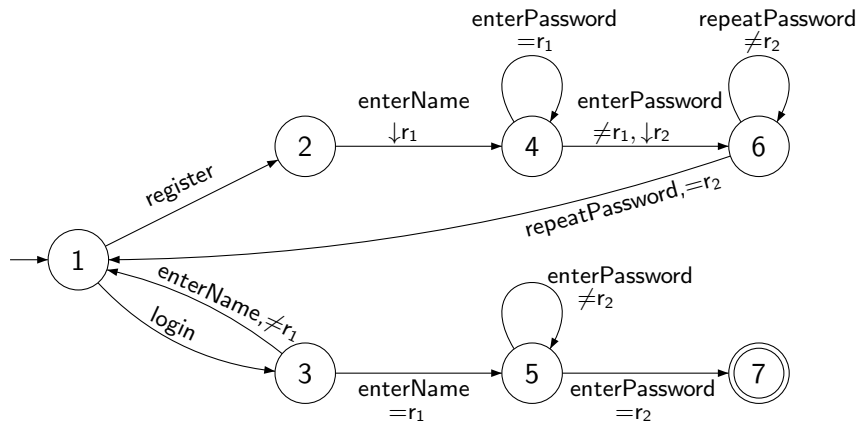
$$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp)$$

Register Automata



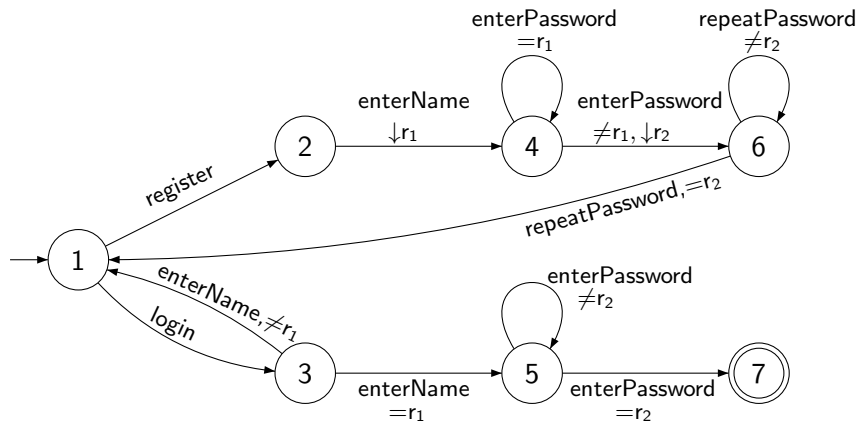
$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp)$

Register Automata



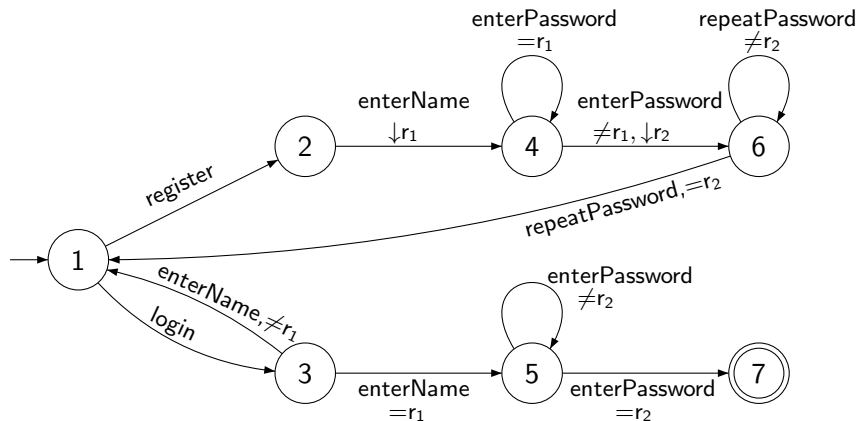
$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$

Register Automata



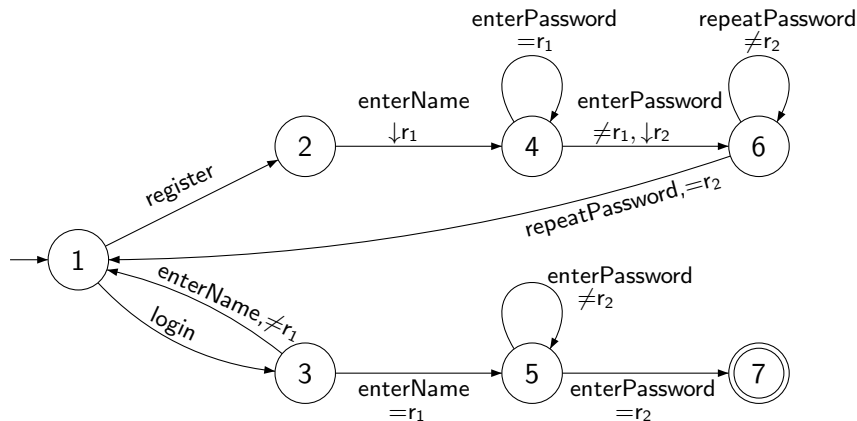
$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc})$

Register Automata



$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc})$

Register Automata

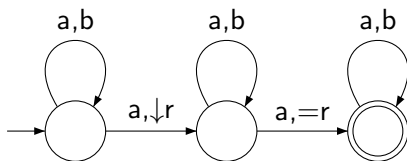


$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

Register Automata

Closure Properties

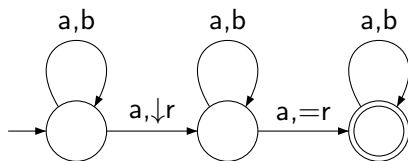
Recognizable data languages are closed under **union**, **intersection**, but not closed under **complement**:



Register Automata

Closure Properties

Recognizable data languages are closed under **union**, **intersection**, but not closed under **complement**:

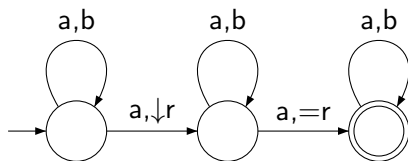


$$L = \{(a_1, d_1) \dots (a_n, d_n) \in (\Sigma \times D)^+ \mid \exists i < j. a_i = a_j = a, d_i = d_j\}$$

Register Automata

Closure Properties

Recognizable data languages are closed under **union**, **intersection**, but not closed under **complement**:



$$L = \{(a_1, d_1) \dots (a_n, d_n) \in (\Sigma \times D)^+ \mid \exists i < j. a_i = a_j = a, d_i = d_j\}$$

$$(\Sigma \times D)^* \setminus L = \{w \in (\Sigma \times D)^* \mid \forall i < j. a_i = a_j = a \Rightarrow d_i \neq d_j\}$$

and cannot be recognized by a register automaton

Register Automata

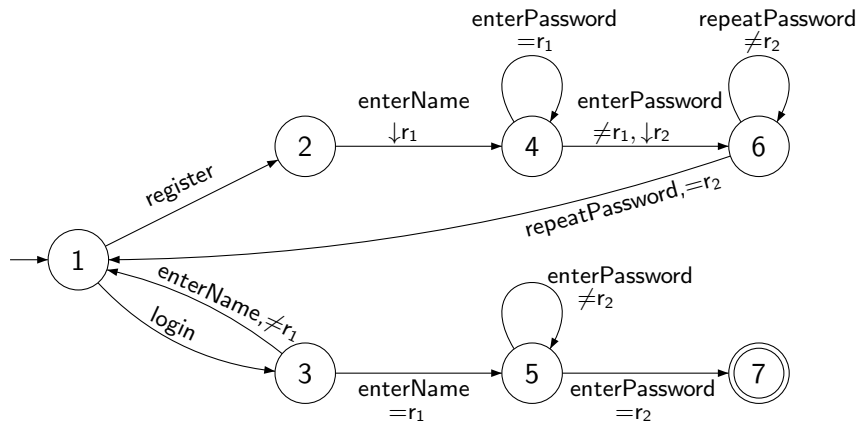
Decision Problems

The Non-Emptiness Problem

Instance: A register automaton \mathcal{A} .

Question: Does \mathcal{A} accept a data word?

Register Automata



$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

Register Automata

Decision Problems: The Non-Emptiness Problem

Abstraction

Given \mathcal{A} , construct an abstract, finite state-transition system $\mathcal{S}_{\mathcal{A}}$:

Register Automata

Decision Problems: The Non-Emptiness Problem

Abstraction

Given \mathcal{A} , construct an abstract, finite state-transition system $\mathcal{S}_{\mathcal{A}}$:

- ▶ Abstract states of the form (s, M, N) , where
 - ▶ s is the current state of \mathcal{A}

Register Automata

Decision Problems: The Non-Emptiness Problem

Abstraction

Given \mathcal{A} , construct an abstract, finite state-transition system $\mathcal{S}_{\mathcal{A}}$:

- ▶ Abstract states of the form (s, M, N) , where
 - ▶ s is the current state of \mathcal{A}
 - ▶ M is a subset of the registers of \mathcal{A} :
 $r \in M \Leftrightarrow$ value of r is equal to the next input datum.

Register Automata

Decision Problems: The Non-Emptiness Problem

Abstraction

Given \mathcal{A} , construct an abstract, finite state-transition system $\mathcal{S}_{\mathcal{A}}$:

- ▶ Abstract states of the form (s, M, N) , where
 - ▶ s is the current state of \mathcal{A}
 - ▶ M is a subset of the registers of \mathcal{A} :
 $r \in M \Leftrightarrow$ value of r is equal to the next input datum.
 - ▶ N is a subset of the cross product of the registers of \mathcal{A} :
 $(r, r') \in N \Leftrightarrow$ values of r and r' are equal

Register Automata

Decision Problems: The Non-Emptiness Problem

Abstraction

Given \mathcal{A} , construct an abstract, finite state-transition system $\mathcal{S}_{\mathcal{A}}$:

- ▶ Abstract states of the form (s, M, N) , where
 - ▶ s is the current state of \mathcal{A}
 - ▶ M is a subset of the registers of \mathcal{A} :
 $r \in M \Leftrightarrow$ value of r is equal to the next input datum.
 - ▶ N is a subset of the cross product of the registers of \mathcal{A} :
 $(r, r') \in N \Leftrightarrow$ values of r and r' are equal
- ▶ Abstract transition relation $(s, M, N) \xRightarrow{a} (s', M', N')$

Register Automata

Decision Problems: The Non-Emptiness Problem

Abstraction

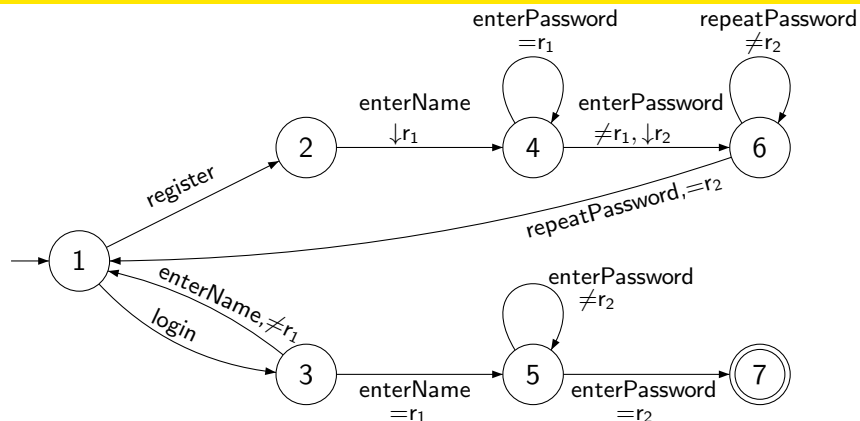
Given \mathcal{A} , construct an abstract, finite state-transition system $\mathcal{S}_{\mathcal{A}}$:

- ▶ Abstract states of the form (s, M, N) , where
 - ▶ s is the current state of \mathcal{A}
 - ▶ M is a subset of the registers of \mathcal{A} :
 $r \in M \Leftrightarrow$ value of r is equal to the next input datum.
 - ▶ N is a subset of the cross product of the registers of \mathcal{A} :
 $(r, r') \in N \Leftrightarrow$ values of r and r' are equal
- ▶ Abstract transition relation $(s, M, N) \xrightarrow{a} (s', M', N')$

There is a successful run in \mathcal{A} , if, and only if, there is a successful run in $\mathcal{S}_{\mathcal{A}}$.

Register Automata

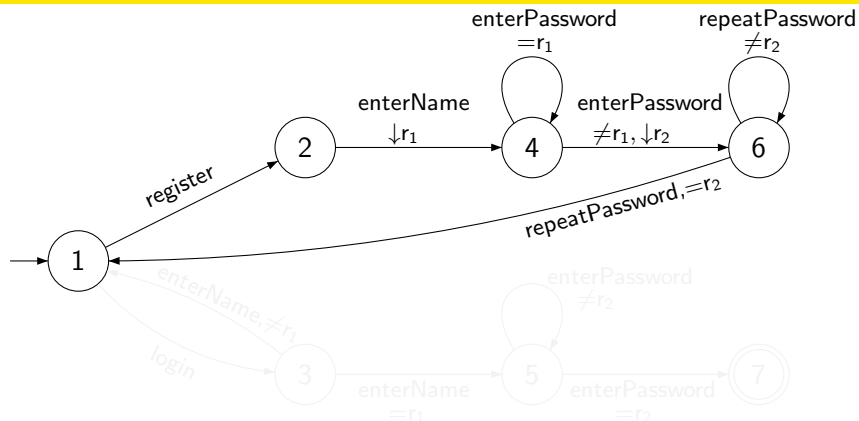
Decision Problems: The Non-Emptiness Problem



$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

Register Automata

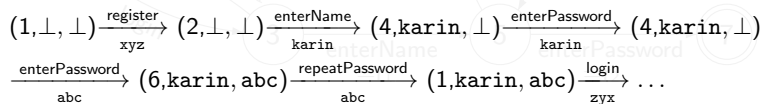
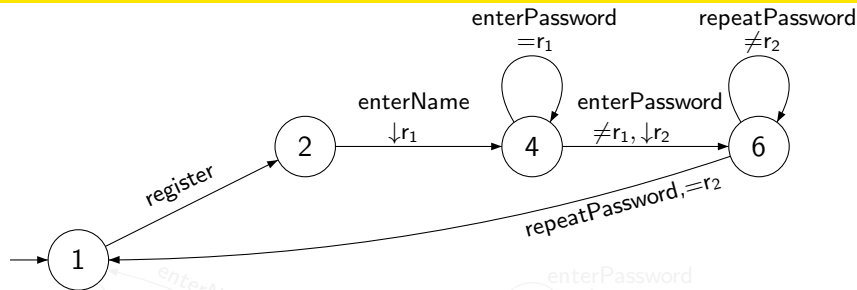
Decision Problems: The Non-Emptiness Problem



$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

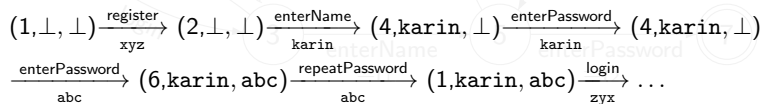
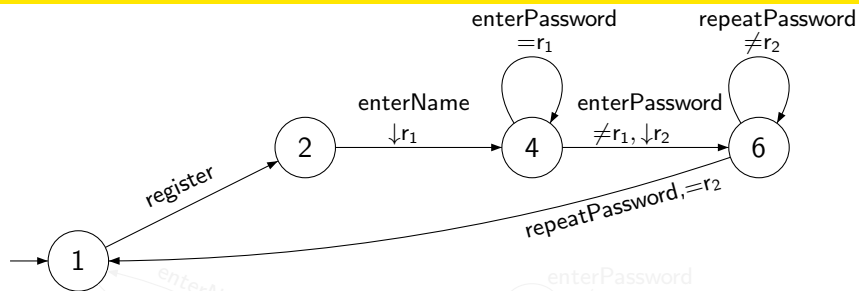
Register Automata

Decision Problems: The Non-Emptiness Problem



Register Automata

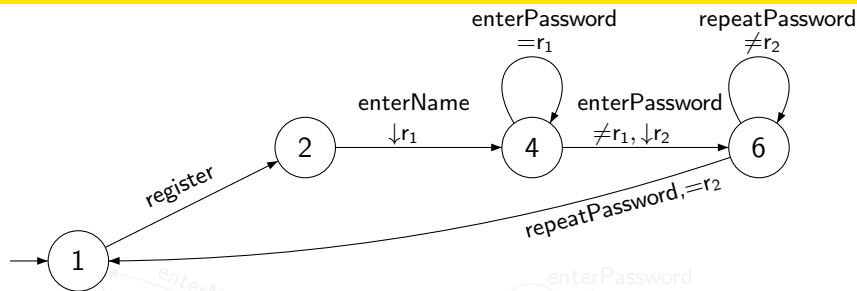
Decision Problems: The Non-Emptiness Problem



$$(1, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{register}} (2, \emptyset, \{r_1, r_2\})$$

Register Automata

Decision Problems: The Non-Emptiness Problem

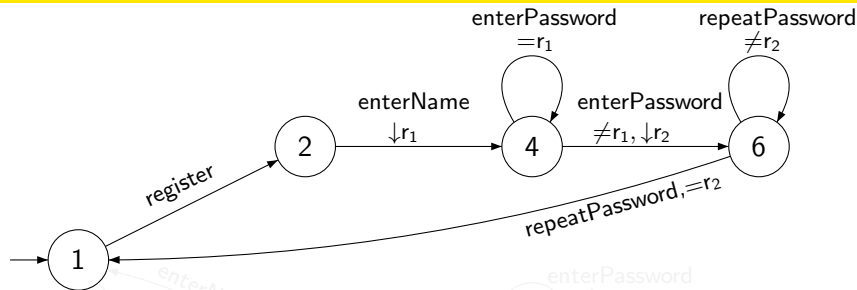


$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

$(1, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{register}} (2, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{enterName}} (4, \{r_1\}, \emptyset)$

Register Automata

Decision Problems: The Non-Emptiness Problem

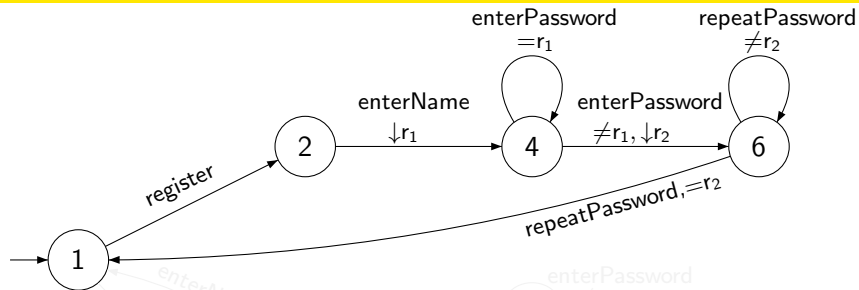


$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

$(1, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{register}} (2, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{enterName}} (4, \{r_1\}, \emptyset) \xrightarrow{\text{enterPassword}} (4, \emptyset, \emptyset)$

Register Automata

Decision Problems: The Non-Emptiness Problem

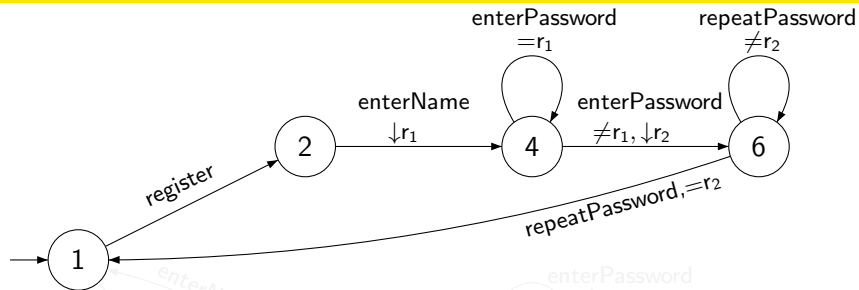


$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

$(1, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{register}} (2, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{enterName}} (4, \{r_1\}, \emptyset) \xrightarrow{\text{enterPassword}} (4, \emptyset, \emptyset)$
 $\xrightarrow{\text{enterPassword}} (6, \{r_2\}, \emptyset)$

Register Automata

Decision Problems: The Non-Emptiness Problem

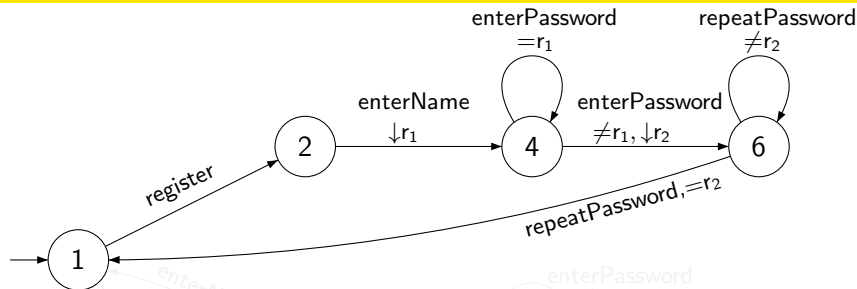


$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

$(1, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{register}} (2, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{enterName}} (4, \{r_1\}, \emptyset) \xrightarrow{\text{enterPassword}} (4, \emptyset, \emptyset)$
 $\xrightarrow{\text{enterPassword}} (6, \{r_2\}, \emptyset) \xrightarrow{\text{repeatPassword}} (1, \emptyset, \emptyset)$

Register Automata

Decision Problems: The Non-Emptiness Problem



$(1, \perp, \perp) \xrightarrow[\text{xyz}]{\text{register}} (2, \perp, \perp) \xrightarrow[\text{karin}]{\text{enterName}} (4, \text{karin}, \perp) \xrightarrow[\text{karin}]{\text{enterPassword}} (4, \text{karin}, \perp)$
 $\xrightarrow[\text{abc}]{\text{enterPassword}} (6, \text{karin}, \text{abc}) \xrightarrow[\text{abc}]{\text{repeatPassword}} (1, \text{karin}, \text{abc}) \xrightarrow[\text{zyx}]{\text{login}} \dots$

$(1, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{register}} (2, \emptyset, \{(r_1, r_2)\}) \xrightarrow{\text{enterName}} (4, \{r_1\}, \emptyset) \xrightarrow{\text{enterPassword}} (4, \emptyset, \emptyset)$
 $\xrightarrow{\text{enterPassword}} (6, \{r_2\}, \emptyset) \xrightarrow{\text{repeatPassword}} (1, \emptyset, \emptyset) \xrightarrow{\text{login}} \dots$

Register Automata

Decision Problems: The Non-Emptiness Problem

Abstraction

Given \mathcal{A} , construct an abstract, finite state-transition system $\mathcal{S}_{\mathcal{A}}$:

- ▶ Abstract states of the form (s, M, N) , where
 - ▶ s is the current state of \mathcal{A}
 - ▶ M is a subset of the registers of \mathcal{A} :
 $r \in M \Leftrightarrow$ value of r is equal to the next input datum.
 - ▶ N is a subset of the cross product of the registers of \mathcal{A} :
 $(r, r') \in N \Leftrightarrow$ values of r and r' are equal
- ▶ Abstract transition relation $(s, M, N) \xrightarrow{a} (s', M', N')$

There is a successful run in \mathcal{A} , if, and only if, there is a successful run in $\mathcal{S}_{\mathcal{A}}$.

Register Automata

Decision Problems

The Non-Emptiness Problem

Instance: A register automaton \mathcal{A} .

Question: Does \mathcal{A} accept a data word?

Theorem (Demri & Lazic, 2008)

The non-emptiness problem is PSPACE-complete.

Register Automata

Decision Problems

The Language Inclusion Problem

Instance: Register automata \mathcal{A}, \mathcal{B} .

Question: Does \mathcal{A} accept a subset of the language accepted by \mathcal{B} ?

Register Automata

Decision Problems

The Language Inclusion Problem

Instance: Register automata \mathcal{A}, \mathcal{B} .

Question: Does \mathcal{A} accept a subset of the language accepted by \mathcal{B} ?

Theorem (Neven, Schwentick & Vianu, 2004)

The language inclusion problem is undecidable.

Freeze LTL

Syntax

$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Freeze LTL

Syntax

$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Semantics

$w = (a_1, d_1) \dots (a_n, d_n), i \in \{1, \dots, n\}, \nu \in (D \cup \{\perp\})^{\mathcal{R}}$

Freeze LTL

Syntax

$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Semantics

$w = (a_1, d_1) \dots (a_n, d_n)$, $i \in \{1, \dots, n\}$, $\nu \in (D \cup \{\perp\})^{\mathcal{R}}$

$$(w, i, \nu) \models a \quad \Leftrightarrow \quad a_i = a$$

$$(w, i, \nu) \models =r \quad \Leftrightarrow$$

$$(w, i, \nu) \models \neg\varphi \quad \Leftrightarrow$$

$$(w, i, \nu) \models \varphi_1 \vee \varphi_2 \quad \Leftrightarrow$$

$$(w, i, \nu) \models X\varphi \quad \Leftrightarrow$$

$$(w, i, \nu) \models \varphi_1 U\varphi_2 \quad \Leftrightarrow$$

$$(w, i, \nu) \models \downarrow r.\varphi \quad \Leftrightarrow$$

Freeze LTL

Syntax

$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Semantics

$w = (a_1, d_1) \dots (a_n, d_n)$, $i \in \{1, \dots, n\}$, $\nu \in (D \cup \{\perp\})^{\mathcal{R}}$

$(w, i, \nu) \models a$	\Leftrightarrow	$a_i = a$
$(w, i, \nu) \models =r$	\Leftrightarrow	$\nu(r) = d_i$
$(w, i, \nu) \models \neg\varphi$	\Leftrightarrow	
$(w, i, \nu) \models \varphi_1 \vee \varphi_2$	\Leftrightarrow	
$(w, i, \nu) \models X\varphi$	\Leftrightarrow	
$(w, i, \nu) \models \varphi_1 U\varphi_2$	\Leftrightarrow	
$(w, i, \nu) \models \downarrow r.\varphi$	\Leftrightarrow	

Freeze LTL

Syntax

$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Semantics

$w = (a_1, d_1) \dots (a_n, d_n)$, $i \in \{1, \dots, n\}$, $\nu \in (D \cup \{\perp\})^{\mathcal{R}}$

$(w, i, \nu) \models a$	\Leftrightarrow	$a_i = a$
$(w, i, \nu) \models =r$	\Leftrightarrow	$\nu(r) = d_i$
$(w, i, \nu) \models \neg\varphi$	\Leftrightarrow	not $(w, i, \nu) \models \varphi$
$(w, i, \nu) \models \varphi_1 \vee \varphi_2$	\Leftrightarrow	
$(w, i, \nu) \models X\varphi$	\Leftrightarrow	
$(w, i, \nu) \models \varphi_1 U\varphi_2$	\Leftrightarrow	
$(w, i, \nu) \models \downarrow r.\varphi$	\Leftrightarrow	

Syntax

$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Semantics

$w = (a_1, d_1) \dots (a_n, d_n)$, $i \in \{1, \dots, n\}$, $\nu \in (D \cup \{\perp\})^{\mathcal{R}}$

$(w, i, \nu) \models a$	\Leftrightarrow	$a_i = a$
$(w, i, \nu) \models =r$	\Leftrightarrow	$\nu(r) = d_i$
$(w, i, \nu) \models \neg\varphi$	\Leftrightarrow	not $(w, i, \nu) \models \varphi$
$(w, i, \nu) \models \varphi_1 \vee \varphi_2$	\Leftrightarrow	$(w, i, \nu) \models \varphi_1$ or $(w, i, \nu) \models \varphi_2$
$(w, i, \nu) \models X\varphi$	\Leftrightarrow	
$(w, i, \nu) \models \varphi_1 U\varphi_2$	\Leftrightarrow	
$(w, i, \nu) \models \downarrow r.\varphi$	\Leftrightarrow	

Freeze LTL

Syntax

$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Semantics

$w = (a_1, d_1) \dots (a_n, d_n)$, $i \in \{1, \dots, n\}$, $\nu \in (D \cup \{\perp\})^{\mathcal{R}}$

$(w, i, \nu) \models a$	\Leftrightarrow	$a_i = a$
$(w, i, \nu) \models =r$	\Leftrightarrow	$\nu(r) = d_i$
$(w, i, \nu) \models \neg\varphi$	\Leftrightarrow	not $(w, i, \nu) \models \varphi$
$(w, i, \nu) \models \varphi_1 \vee \varphi_2$	\Leftrightarrow	$(w, i, \nu) \models \varphi_1$ or $(w, i, \nu) \models \varphi_2$
$(w, i, \nu) \models X\varphi$	\Leftrightarrow	$i + 1 \leq n$, $(w, i + 1, \nu) \models \varphi$
$(w, i, \nu) \models \varphi_1 U\varphi_2$	\Leftrightarrow	
$(w, i, \nu) \models \downarrow r.\varphi$	\Leftrightarrow	

Syntax

$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Semantics

$w = (a_1, d_1) \dots (a_n, d_n)$, $i \in \{1, \dots, n\}$, $\nu \in (D \cup \{\perp\})^{\mathcal{R}}$

$(w, i, \nu) \models a$	\Leftrightarrow	$a_i = a$
$(w, i, \nu) \models =r$	\Leftrightarrow	$\nu(r) = d_i$
$(w, i, \nu) \models \neg\varphi$	\Leftrightarrow	not $(w, i, \nu) \models \varphi$
$(w, i, \nu) \models \varphi_1 \vee \varphi_2$	\Leftrightarrow	$(w, i, \nu) \models \varphi_1$ or $(w, i, \nu) \models \varphi_2$
$(w, i, \nu) \models X\varphi$	\Leftrightarrow	$i + 1 \leq n$, $(w, i + 1, \nu) \models \varphi$
$(w, i, \nu) \models \varphi_1 U\varphi_2$	\Leftrightarrow	$\exists i < j \leq n. (w, j, \nu) \models \varphi_2, \forall i < k < j. (w, k, \nu) \models \varphi_1$
$(w, i, \nu) \models \downarrow r.\varphi$	\Leftrightarrow	

Syntax

$$\varphi ::= a \mid =r \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \downarrow r.\varphi$$

where $a \in \Sigma$ and r is a register variable in the set \mathcal{R} or registers

Semantics

$$w = (a_1, d_1) \dots (a_n, d_n), i \in \{1, \dots, n\}, \nu \in (D \cup \{\perp\})^{\mathcal{R}}$$

$(w, i, \nu) \models a$	\Leftrightarrow	$a_i = a$
$(w, i, \nu) \models =r$	\Leftrightarrow	$\nu(r) = d_i$
$(w, i, \nu) \models \neg\varphi$	\Leftrightarrow	not $(w, i, \nu) \models \varphi$
$(w, i, \nu) \models \varphi_1 \vee \varphi_2$	\Leftrightarrow	$(w, i, \nu) \models \varphi_1$ or $(w, i, \nu) \models \varphi_2$
$(w, i, \nu) \models X\varphi$	\Leftrightarrow	$i + 1 \leq n, (w, i + 1, \nu) \models \varphi$
$(w, i, \nu) \models \varphi_1 U\varphi_2$	\Leftrightarrow	$\exists i < j \leq n. (w, j, \nu) \models \varphi_2, \forall i < k < j. (w, k, \nu) \models \varphi_1$
$(w, i, \nu) \models \downarrow r.\varphi$	\Leftrightarrow	$(w, i, \nu[r := d_i]) \models \varphi$

Freeze LTL

Examples

Let $w = (a_1, d_1)(a_2, d_2)(a_3, d_3)(a_4, d_4)(a_5, d_5)$ be a data word, let $\nu \in D \cup \{\perp\}$.

$$(w, 1, \nu) \models a \quad \Leftrightarrow \quad a_1 = a$$

Freeze LTL

Examples

Let $w = (a_1, d_1)(a_2, d_2)(a_3, d_3)(a_4, d_4)(a_5, d_5)$ be a data word, let $\nu \in D \cup \{\perp\}$.

$$(w, 1, \nu) \models a$$

$$\Leftrightarrow a_1 = a$$

$$(w, 1, \nu) \models \downarrow r. = r$$

$$\Leftrightarrow \text{always true}$$

Freeze LTL

Examples

Let $w = (a_1, d_1)(a_2, d_2)(a_3, d_3)(a_4, d_4)(a_5, d_5)$ be a data word, let $\nu \in D \cup \{\perp\}$.

$$(w, 1, \nu) \models a \quad \Leftrightarrow \quad a_1 = a$$

$$(w, 1, \nu) \models \downarrow r. = r \quad \Leftrightarrow \quad \text{always true}$$

$$(w, 4, \nu) \models (b \wedge Xa) \rightarrow \downarrow r. X = r \quad \Leftrightarrow \quad a_4 = b, a_5 = a \Rightarrow d_4 = d_5$$

Freeze LTL

Examples

Let $w = (a_1, d_1)(a_2, d_2)(a_3, d_3)(a_4, d_4)(a_5, d_5)$ be a data word, let $\nu \in D \cup \{\perp\}$.

$(w, 1, \nu) \models a$	$\Leftrightarrow a_1 = a$
$(w, 1, \nu) \models \downarrow r. = r$	\Leftrightarrow always true
$(w, 4, \nu) \models (b \wedge Xa) \rightarrow \downarrow r. X = r$	$\Leftrightarrow a_4 = b, a_5 = a \Rightarrow d_4 = d_5$
$(w, 1, \nu) \models \downarrow r. bU(b \wedge \neq r)$	$\Leftrightarrow \exists 1 < i \leq 5. a_i = b, d_i \neq d_1,$ $\forall 1 < j < i. a_j = b$

Freeze LTL

Two useful abbreviations

Definition

$$F\varphi := \text{true} \cup \varphi$$

$$G\varphi := \neg F\neg\varphi$$

Semantics

$$w = (a_1, d_1) \dots (a_n, d_n), i \in \{1, \dots, n\}, \nu \in (D \cup \{\perp\})^{\mathcal{R}}$$

$$(w, i, \nu) \models F\varphi \quad \Leftrightarrow \quad \exists 1 < i \leq n. (w, i, \nu) \models \varphi$$

$$(w, i, \nu) \models G\varphi \quad \Leftrightarrow \quad \forall 1 < i \leq n. (w, i, \nu) \models \varphi$$

Freeze LTL

Decision Problems

The Satisfiability Problem

Instance: Freeze LTL formula φ .

Question: Does there exist a data word such that $(w, 1, \{\perp\}^{\mathcal{R}}) \models \varphi$?

Freeze LTL

Decision Problems

The Satisfiability Problem

Instance: Freeze LTL formula φ .

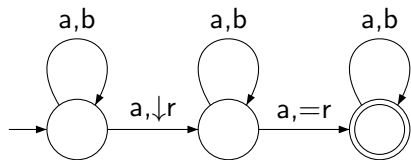
Question: Does there exist a data word such that $(w, 1, \{\perp\}^{\mathcal{R}}) \models \varphi$?

Can we use an automata-based approach to show decidability?

Freeze LTL

Can Formulas be translated to equivalent Register Automata?

Recognizable data languages are closed under **union**, **intersection**, but not closed under **complement**:



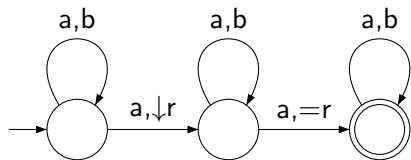
$$L = \{(a_1, d_1) \dots (a_n, d_n) \in (\Sigma \times D)^+ \mid \exists i < j. a_i = a_j = a, d_i = d_j\}$$

$(\Sigma \times D)^* \setminus L$ cannot be recognized by a register automaton

Freeze LTL

Can Formulas be translated to equivalent Register Automata?

Recognizable data languages are closed under **union**, **intersection**, but not closed under **complement**:



$$L = \{(a_1, d_1) \dots (a_n, d_n) \in (\Sigma \times D)^+ \mid \exists i < j. a_i = a_j = a, d_i = d_j\}$$

$(\Sigma \times D)^* \setminus L$ cannot be recognized by a register automaton

Hence, the formula $\neg F(a \wedge \downarrow r. F(a \wedge = r))$ cannot be translated into an equivalent register automaton

Freeze LTL

Decision Problems

The Satisfiability Problem

Instance: Freeze LTL formula φ .

Question: Does there exist a data word such that $(w, 1, \{\perp\}^{\mathcal{R}}) \models \varphi$?

Theorem (Demri & Lazic, 2008)

The satisfiability problem is undecidable.

Freeze LTL

Decision Problems

The Satisfiability Problem

Instance: Freeze LTL formula φ .

Question: Does there exist a data word such that $(w, 1, \{\perp\}^{\mathcal{R}}) \models \varphi$?

Theorem (Demri & Lazic, 2008)

The satisfiability problem is undecidable. But: the satisfiability problem is decidable for Freeze LTL with at most one register.

Freeze LTL

Decision Problems

The Satisfiability Problem

Instance: Freeze LTL formula φ .

Question: Does there exist a data word such that $(w, 1, \{\perp\}^{\mathcal{R}}) \models \varphi$?

Theorem (Demri & Lazic, 2008)

The satisfiability problem is undecidable. But: the satisfiability problem is decidable for Freeze LTL with at most one register.

Proof idea: Automata-based approach!

- ▶ Every Freeze LTL formula with at most one register can be translated into an equivalent **alternating** single-register automaton.

The Satisfiability Problem

Instance: Freeze LTL formula φ .

Question: Does there exist a data word such that $(w, 1, \{\perp\}^{\mathcal{R}}) \models \varphi$?

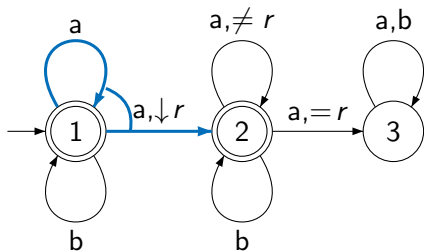
Theorem (Demri & Lazic, 2008)

The satisfiability problem is undecidable. But: the satisfiability problem is decidable for Freeze LTL with at most one register.

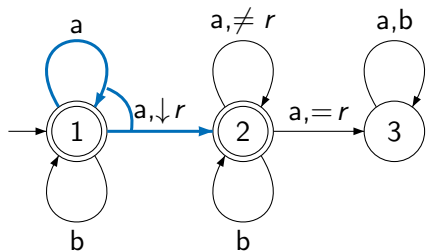
Proof idea: Automata-based approach!

- ▶ Every Freeze LTL formula with at most one register can be translated into an equivalent **alternating** single-register automaton.
- ▶ The non-emptiness problem for alternating single-register automata is decidable.

Alternating Single-Register Automata

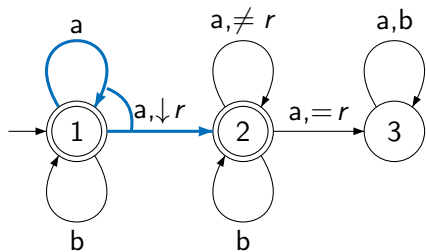


Alternating Single-Register Automata



	a	b
1	$1 \wedge \downarrow r.2$	1
2	$(2 \wedge \neq r) \vee (3 \wedge = r)$	2
3	3	3

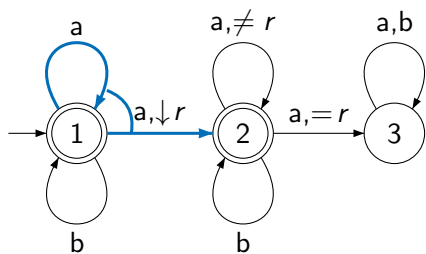
Alternating Single-Register Automata



	a	b
1	$1 \wedge \downarrow r.2$	1
2	$(2 \wedge \neq r) \vee (3 \wedge = r)$	2
3	3	3

$$\{(1, \perp)\} \xrightarrow[d_1]{b}$$

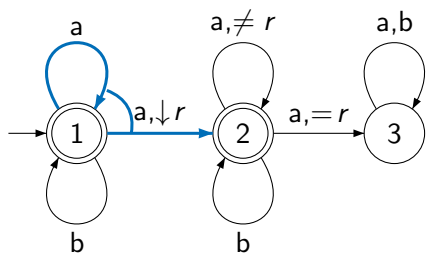
Alternating Single-Register Automata



	a	b
1	$1 \wedge \downarrow r.2$	1
2	$(2 \wedge \neq r) \vee (3 \wedge = r)$	2
3	3	3

$$\{(1, \perp)\} \xrightarrow{d_1^b} \{(1, \perp)\} \xrightarrow{d_2^a}$$

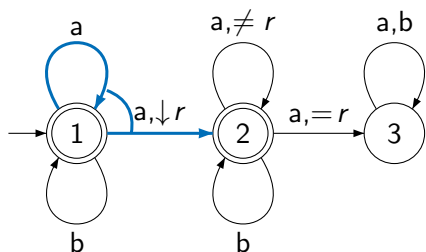
Alternating Single-Register Automata



	a	b
1	$1 \wedge \downarrow r.2$	1
2	$(2 \wedge \neq r) \vee (3 \wedge = r)$	2
3	3	3

$$\{(1, \perp)\} \xrightarrow{b}_{d_1} \{(1, \perp)\} \xrightarrow{a}_{d_2} \{(1, \perp), (2, d_2)\} \xrightarrow{a}_{d_3}$$

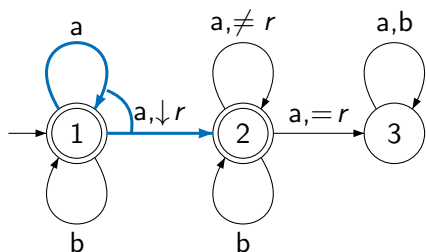
Alternating Single-Register Automata



	a	b
1	$1 \wedge \downarrow r.2$	1
2	$(2 \wedge \neq r) \vee (3 \wedge = r)$	2
3	3	3

$$\{(1, \perp)\} \xrightarrow{b}_{d_1} \{(1, \perp)\} \xrightarrow{a}_{d_2} \{(1, \perp), (2, d_2)\} \xrightarrow{a}_{d_3} \{(1, \perp), (2, d_2), (2, d_3)\} \xrightarrow{b}_{d_2}$$

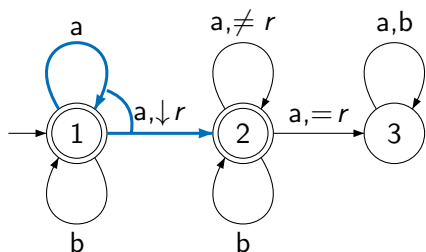
Alternating Single-Register Automata



	a	b
1	$1 \wedge \downarrow r.2$	1
2	$(2 \wedge \neq r) \vee (3 \wedge = r)$	2
3	3	3

$$\begin{aligned}
 \{(1, \perp)\} &\xrightarrow{d_1^b} \{(1, \perp)\} \xrightarrow{d_2^a} \{(1, \perp), (2, d_2)\} \xrightarrow{d_3^a} \{(1, \perp), (2, d_2), (2, d_3)\} \xrightarrow{d_2^b} \\
 &\{(1, \perp), (2, d_2), (2, d_3)\} \xrightarrow{d_1^a}
 \end{aligned}$$

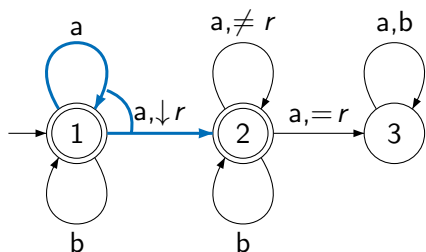
Alternating Single-Register Automata



	a	b
1	$1 \wedge \downarrow r.2$	1
2	$(2 \wedge \neq r) \vee (3 \wedge = r)$	2
3	3	3

$$\begin{aligned} \{(1, \perp)\} &\xrightarrow{d_1^b} \{(1, \perp)\} \xrightarrow{d_2^a} \{(1, \perp), (2, d_2)\} \xrightarrow{d_3^a} \{(1, \perp), (2, d_2), (2, d_3)\} \xrightarrow{d_2^b} \\ &\{(1, \perp), (2, d_2), (2, d_3)\} \xrightarrow{d_1^a} \{(1, \perp), (2, d_2), (2, d_3), (2, d_1)\} \xrightarrow{d_3^a} \end{aligned}$$

Alternating Single-Register Automata



	a	b
1	$1 \wedge \downarrow r.2$	1
2	$(2 \wedge \neq r) \vee (3 \wedge = r)$	2
3	3	3

$$\begin{aligned}
 \{(1, \perp)\} &\xrightarrow{d_1^b} \{(1, \perp)\} \xrightarrow{d_2^a} \{(1, \perp), (2, d_2)\} \xrightarrow{d_3^a} \{(1, \perp), (2, d_2), (2, d_3)\} \xrightarrow{d_2^b} \\
 \{(1, \perp), (2, d_2), (2, d_3)\} &\xrightarrow{d_1^a} \{(1, \perp), (2, d_2), (2, d_3), (2, d_1)\} \xrightarrow{d_3^a} \\
 \{(1, \perp), (2, d_2), (2, d_3), (2, d_1), (3, d_3)\} &\xrightarrow{d_4^a} \dots
 \end{aligned}$$

Alternating Single-Register Automata

Decision Problems

Theorem (Demri & Lazic, 2008)

The non-emptiness problem for alternating single-register automata is decidable.

Alternating Single-Register Automata

Decision Problems

Theorem (Demri & Lazic, 2008)

The non-emptiness problem for alternating single-register automata is decidable.

Two proof variants

- ▶ Reduction to the non-emptiness problem of counter automata with insertion errors
- ▶ The semantical graph is a well-structured transition system

Alternating Single-Register Automata

Decision Problems

Theorem (Demri & Lazic, 2008)

The non-emptiness problem for alternating single-register automata is decidable.

Two proof variants

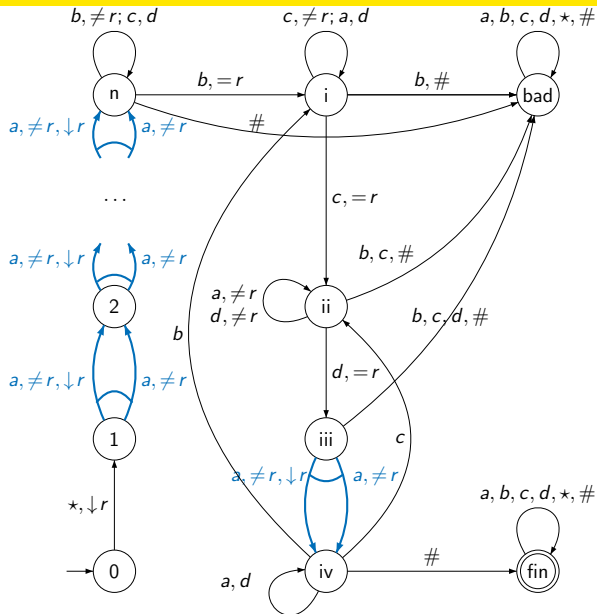
- ▶ Reduction to the non-emptiness problem of counter automata with insertion errors
- ▶ The semantical graph is a well-structured transition system

Bad complexity!

The problem is Ackermann-complete.

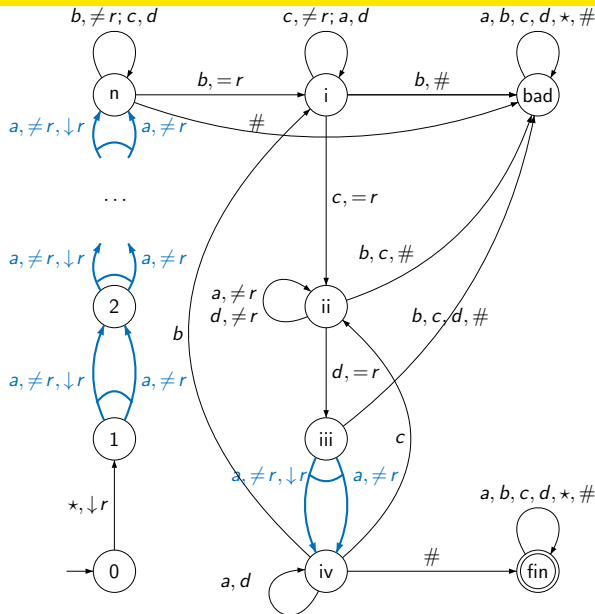
Alternating Single-Register Automata

Quiz: Which fast-growing function does this family of automata simulate?



Alternating Single-Register Automata

Quiz: Which fast-growing function does this family of automata simulate?



- $\{(0, \perp)\}$
- $\downarrow \ast, 1$
- $\{(1, 1)\}$
- $\downarrow a, 2$
- $\{(2, 1), (2, 2)\}$
- $\downarrow a, 3$
- \dots

- ▶ Demri & Lazic: *LTL with the Freeze Quantifier and Register Automata*. ACM Transactions on Computational Logic, 2008.
- ▶ Neven, Schwentick & Vianu: *Finite State Machines for Strings over Infinite Alphabets*. ACM Transactions on Computational Logic, 2001.
- ▶ Figueira: *Alternating register automata on finite words and trees*. Logical Methods in Computer Science, 2012.