Alternating Data Automata

XIAO XU

VERIMAG [CNRS/UGA]

Index

Inclusion Problems

Inclusion Problems

Model and Specifications

Inclusion Problems

Model and Specifications

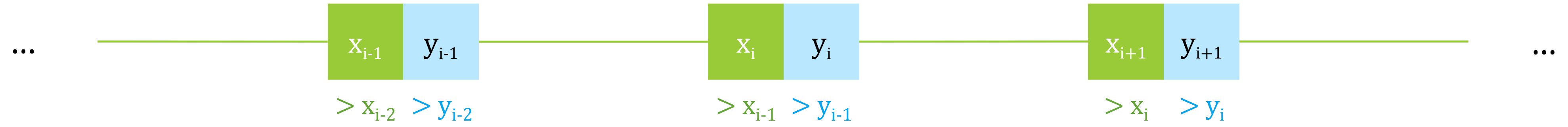
<u>Model 1</u> $x_0 = 1, y_0 = 1, \text{for all } i > 0: x_i = y_{i\text{-}1}, y_i = x_{i\text{-}1} + y_{i\text{-}1}$

Inclusion Problems

Model and Specifications

<u>Model 1</u> $x_0 = 1, y_0 = 1, \text{for all } i > 0: x_i = y_{i-1}, y_i = x_{i-1} + y_{i-1}$

<u>Model 2</u> $x_0 = 1, y_0 = 1, \text{for all } i > 0: x_i > x_{i-1}, y_i > y_{i-1}$



Inclusion Problems

Model and Specifications

Model 1 $x_0 = 1, y_0 = 1$, for all $i > 0$: $x_i = y_{i-1}, y_i = x_{i-1} + y_{i-1}$ Specification1

1 1 — 1 2 — 2 3 — 3 5 — 5 8

Model 2 $x_0 = 1, y_0 = 1$, for all $i > 0$: $x_i > x_{i-1}, y_i > y_{i-1}$ Specification2

… — x_{i-1} y_{i-1} — x_i y_i — x_{i+1} y_{i+1} — …

$> x_{i-2}$ $> y_{i-2}$ $> x_{i-1}$ $> y_{i-1}$ $> x_i$ $> y_i$

Question Is every model that meets specification1 also meets specification2?

Inclusion Problems

Inclusion and Emptiness

Inclusion Problems

Inclusion and Emptiness

Situation:
Programs using container libraries, for instance ordered queues and arrays.

Inclusion Problems

Inclusion and Emptiness

Situation:
Programs using container libraries, for instance ordered queues and arrays.

One possible solution:
Using finite automata, reduce the problem into a language inclusion problem.

Model A → Automaton A
Model B → Automaton B

Problem → whether $\mathrm{L(A)} \subseteq \mathrm{L(B)}$

Inclusion Problems

Inclusion and Emptiness

Situation:
Programs using container libraries, for instance ordered queues and arrays.

One possible solution:
Using finite automata, reduce the problem into a language inclusion problem.

Model A → Automaton A
Model B → Automaton B

Problem → whether $\mathrm{L}(\mathrm{A}) \subseteq \mathrm{L}(\mathrm{B})$ → whether $\mathrm{L}(\mathrm{A}) \cap \mathrm{L}(\overline{\mathrm{B}}) = \emptyset$ <u>emptiness checking</u>

Inclusion Problems

Inclusion and Emptiness

Situation:
Programs using container libraries, for instance ordered queues and arrays.

One possible solution:
Using finite automata, reduce the problem into a language inclusion problem.

Model A → Automaton A
Model B → Automaton B

Problem → whether $\mathrm{L(A)} \subseteq \mathrm{L(B)}$ → whether $\mathrm{L(A)} \cap \mathrm{L}(\overline{\mathrm{B}}) = \emptyset$ <u>emptiness checking</u>

<u>Difficulty</u> Infinite Alphabets

Inclusion Problems

Inclusion and Emptiness

Situation:
Programs using container libraries, for instance ordered queues and arrays.

One possible solution:
Using finite automata, reduce the problem into a language inclusion problem.

Model A → Automaton A
Model B → Automaton B

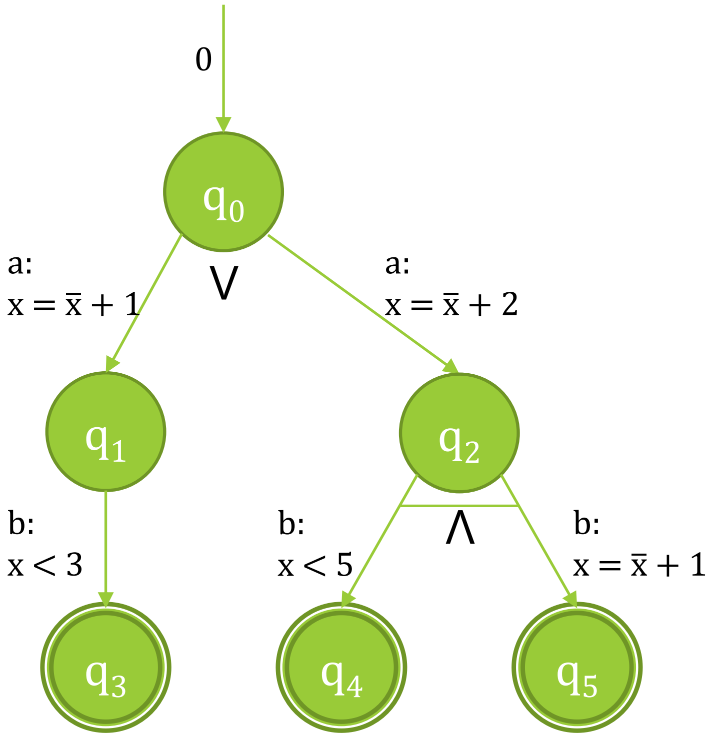
Problem → whether $\mathrm{L}(\mathrm{A}) \subseteq \mathrm{L}(\mathrm{B})$ → whether $\mathrm{L}(\mathrm{A}) \cap \mathrm{L}(\overline{\mathrm{B}}) = \emptyset$ <u>emptiness checking</u>

<u>Difficulty</u> Infinite Alphabets

<u>Solution</u> Alternating Data Automata

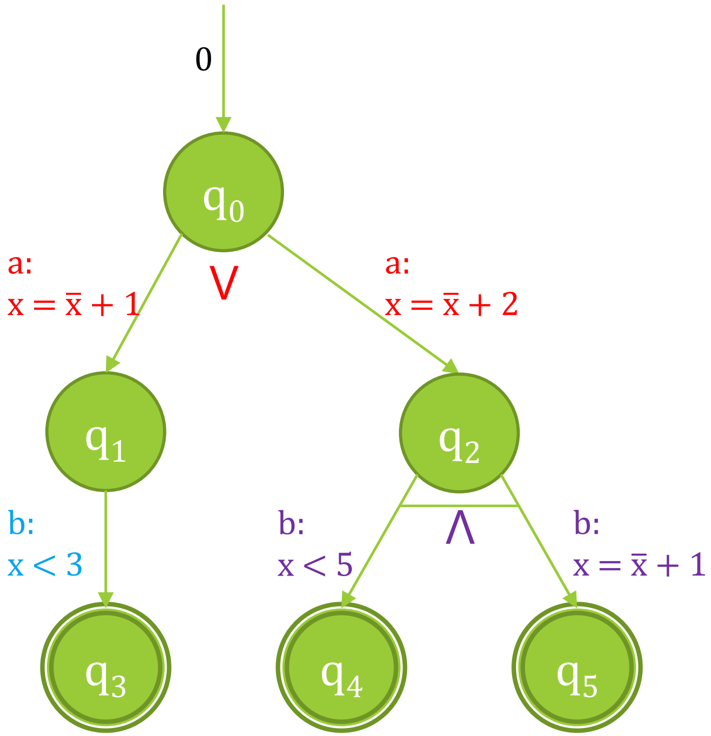
Alternating Data Automata

Alternating Data Automata
Definition



Alternating Data Automata

Definition



$A = < X, Q, \iota, F, \Delta >$

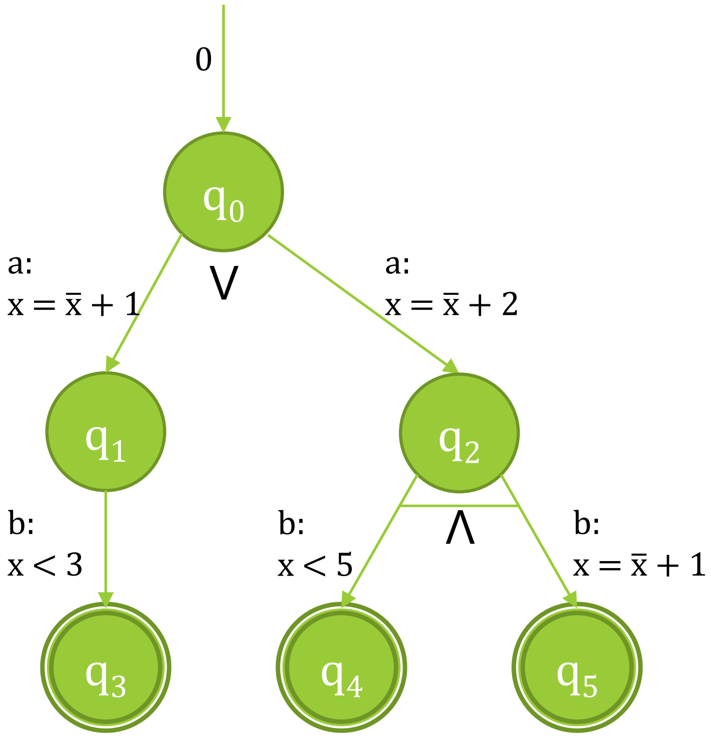
- $X = \{x\}$; $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$; $\iota = q_0 \wedge x_0 = 0$; $F = \{q_3, q_4, q_5\}$;
- Δ is presented as the following table (must be COMPLETE)

	a	b
q_0	$q_1 \wedge x = 1 \vee q_2 \wedge x = 2$	false
q_1	false	$q_3 \wedge x < 3$
q_2	false	$q_4 \wedge x < 5 \wedge q_5 \wedge x = \bar{x} + 1$
q_3	false	false
q_4	false	false
q_5	false	false

where $\bar{x}$ is the previous value of x

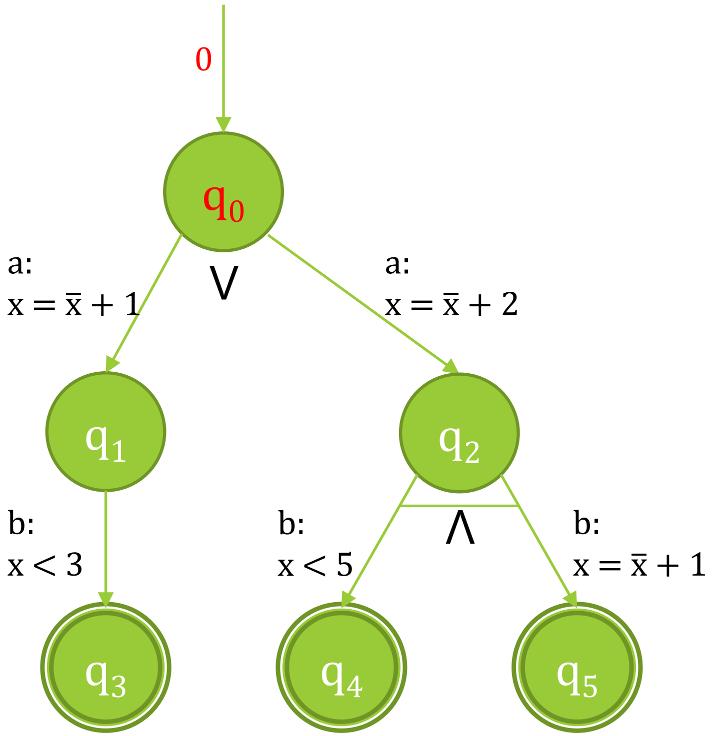
Alternating Data Automata

Concrete Post Image



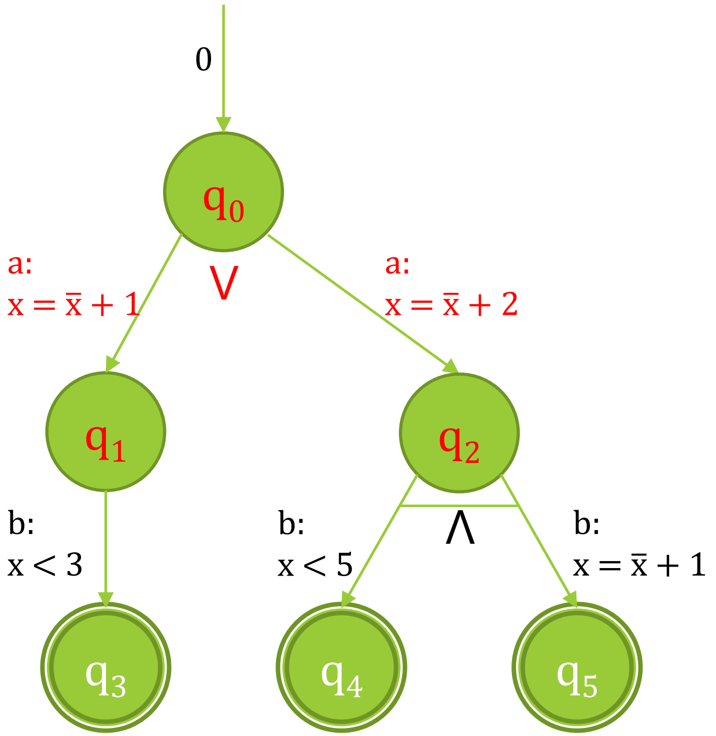
Alternating Data Automata
Concrete Post Image

$q_0 \land x_0 = 0$



Alternating Data Automata

Concrete Post Image

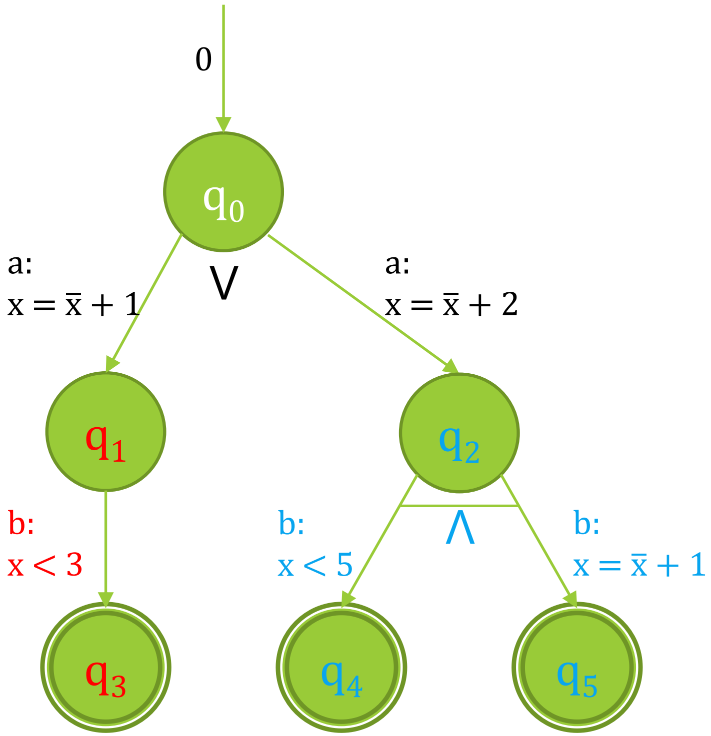


$q_0 \wedge x_0 = 0$

$$
\begin{aligned}
& \mathrm{POST}(q_0 \wedge x_0 = 0, a) \\
= \; & q_0 \wedge x_0 = 0 \wedge (q_0 \rightarrow (q_1 \wedge x_1 = x_0 + 1 \vee q_2 \wedge x_1 = x_0 + 2)) \\
= \; & x_0 = 0 \wedge (q_1 \wedge x_1 = 1 \vee q_2 \wedge x_1 = 2)
\end{aligned}
$$

Alternating Data Automata

Concrete Post Image



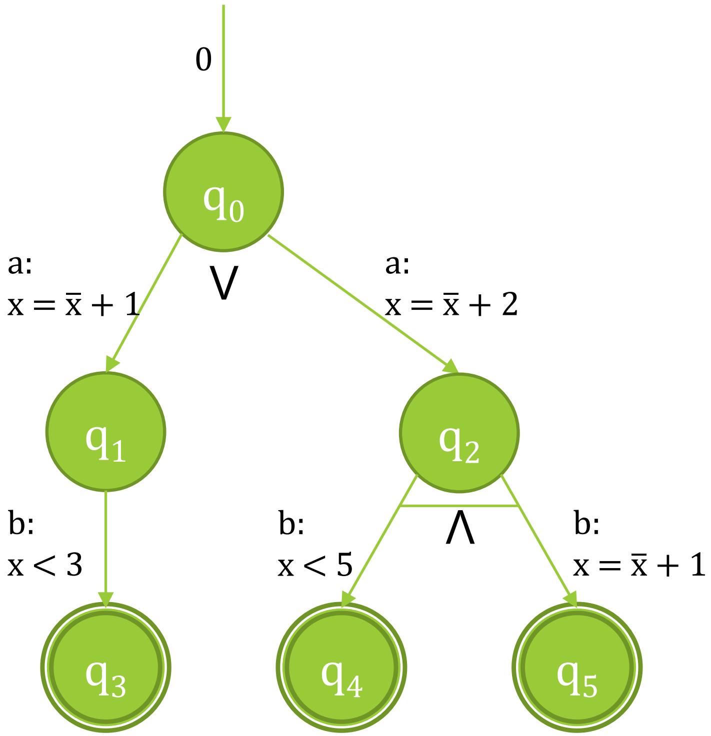
$q_0 \wedge x_0 = 0$

$$
\begin{aligned}
& \mathrm{POST}(q_0 \wedge x_0 = 0, a) \\
= \ & q_0 \wedge x_0 = 0 \wedge (q_0 \rightarrow (q_1 \wedge x_1 = x_0 + 1 \vee q_2 \wedge x_1 = x_0 + 2)) \\
= \ & x_0 = 0 \wedge (q_1 \wedge x_1 = 1 \vee q_2 \wedge x_1 = 2)
\end{aligned}
$$

$$
\begin{aligned}
& \mathrm{POST}(x_0 = 0 \wedge (q_1 \wedge x_1 = 1 \vee q_2 \wedge x_1 = 2), b) \\
= \ & x_0 = 0 \wedge (q_1 \wedge x_1 = 1 \vee q_2 \wedge x_1 = 2) \\
& \wedge (q_1 \rightarrow (q_3 \wedge x_2 < 3)) \wedge (q_2 \rightarrow (q_4 \wedge x_2 < 5 \wedge q_5 \wedge x_2 = x_1 + 1)) \\
= \ & x_0 = 0 \wedge (q_3 \wedge x_2 < 3 \wedge x_1 = 1 \vee q_4 \wedge q_5 \wedge x_2 = 3 \wedge x_1 = 2)
\end{aligned}
$$

Alternating Data Automata

Concrete Post Image



$q_0 \wedge x_0 = 0$

$$
\begin{aligned}
& \mathrm{POST}(q_0 \wedge x_0 = 0, a) \\
= \quad & q_0 \wedge x_0 = 0 \wedge (q_0 \rightarrow (q_1 \wedge x_1 = x_0 + 1 \vee q_2 \wedge x_1 = x_0 + 2)) \\
= \quad & x_0 = 0 \wedge (q_1 \wedge x_1 = 1 \vee q_2 \wedge x_1 = 2)
\end{aligned}
$$

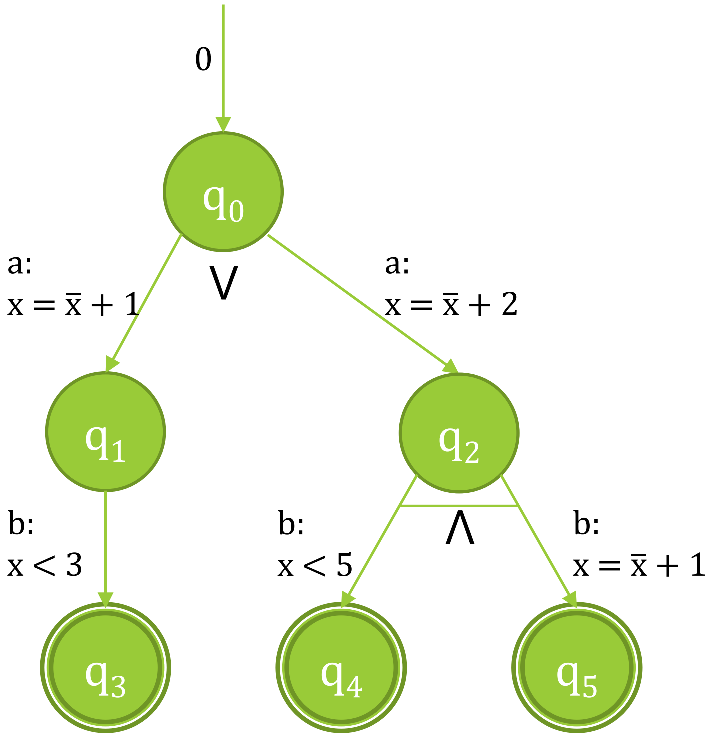
$$
\begin{aligned}
& \mathrm{POST}(x_0 = 0 \wedge (q_1 \wedge x_1 = 1 \vee q_2 \wedge x_1 = 2), b) \\
= \quad & x_0 = 0 \wedge (q_1 \wedge x_1 = 1 \vee q_2 \wedge x_1 = 2) \\
& \wedge (q_1 \rightarrow (q_3 \wedge x_2 < 3)) \wedge (q_2 \rightarrow (q_4 \wedge x_2 < 5 \wedge q_5 \wedge x_2 = x_1 + 1)) \\
= \quad & x_0 = 0 \wedge (q_3 \wedge x_2 < 3 \wedge x_1 = 1 \vee q_4 \wedge q_5 \wedge x_2 = 3 \wedge x_1 = 2)
\end{aligned}
$$

Hence:

$$
\begin{aligned}
& \mathrm{POST}(q_0 \wedge x_0 = 0, ab) \\
= \quad & x_0 = 0 \wedge (q_3 \wedge x_2 < 3 \wedge x_1 = 1 \vee q_4 \wedge q_5 \wedge x_2 = 3 \wedge x_1 = 2)
\end{aligned}
$$

Alternating Data Automata

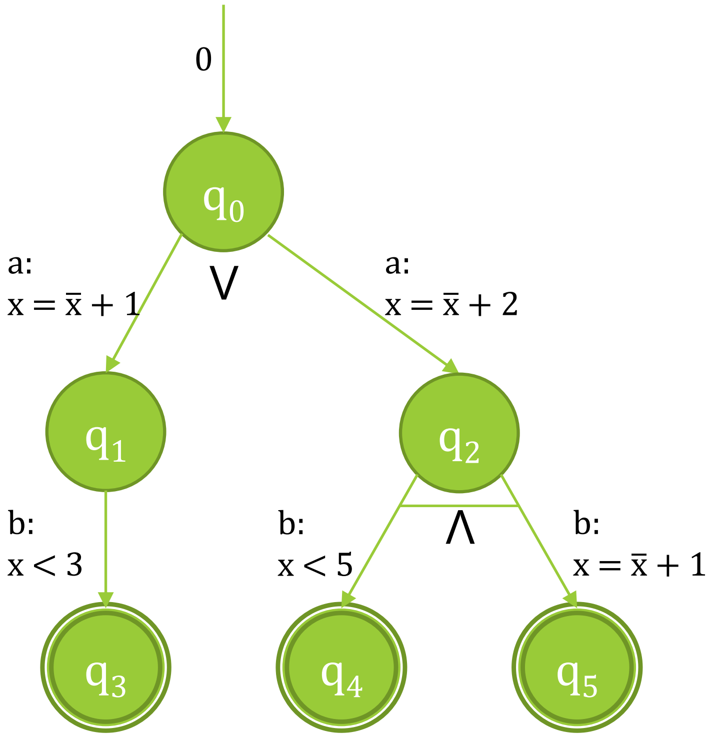
Accepting Word



Data Words $w \in (\Sigma \times D^X)^*$

Alternating Data Automata

Accepting Word

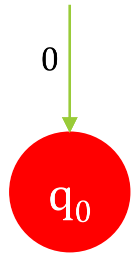


Data Words $w \in (\Sigma \times D^X)^*$

For example, the left alternating data automaton accepts the data word:
$(a, 2)\ (b, 3)$

Alternating Data Automata

Accepting Word



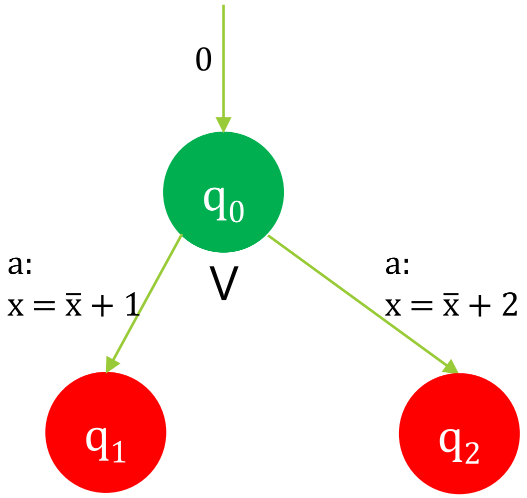
$x_0 = 0$

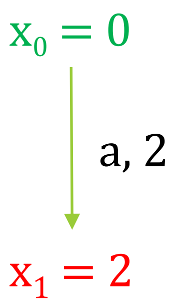
Data Words $w \in (\Sigma \times D^X)^*$

For example, the left alternating data automaton accepts the data word:
$(a, 2)\ (b, 3)$

Previous Data	Current Data
/	$x_0 = 0$

Alternating Data Automata
Accepting Word





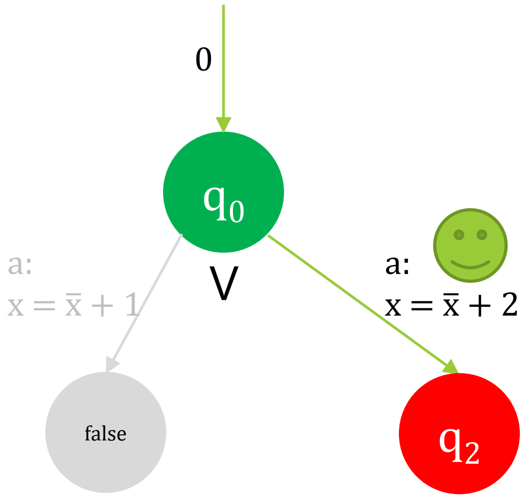
Data Words $w \in (\Sigma \times D^X)^*$

For example, the left alternating data automaton accepts the data word:
$(a, 2)\ (b, 3)$

Previous Data	**Current Data**
$x_0 = 0$	$x_1 = 2$

Alternating Data Automata

Accepting Word



$$x_0 = 0 \xrightarrow{a,\ 2} x_1 = 2$$

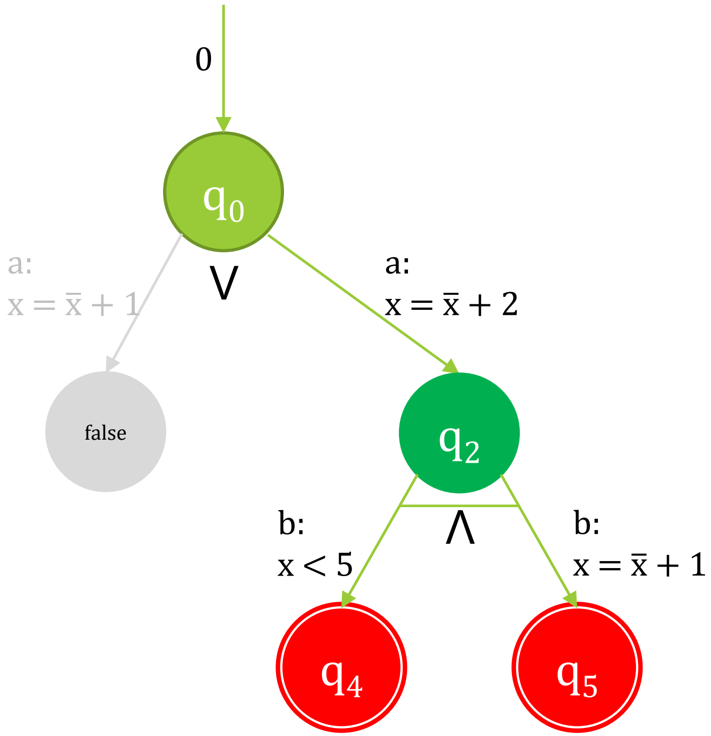
Data Words $w \in (\Sigma \times D^X)^*$

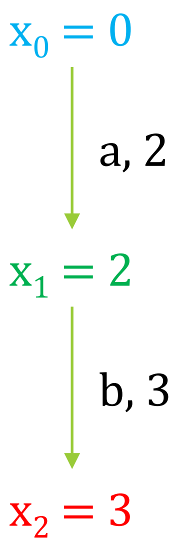
For example, the left alternating data automaton accepts the data word:
$(a, 2)\ (b, 3)$

Previous Data	**Current Data**
$x_0 = 0$	$x_1 = 2$

Alternating Data Automata

Accepting Word





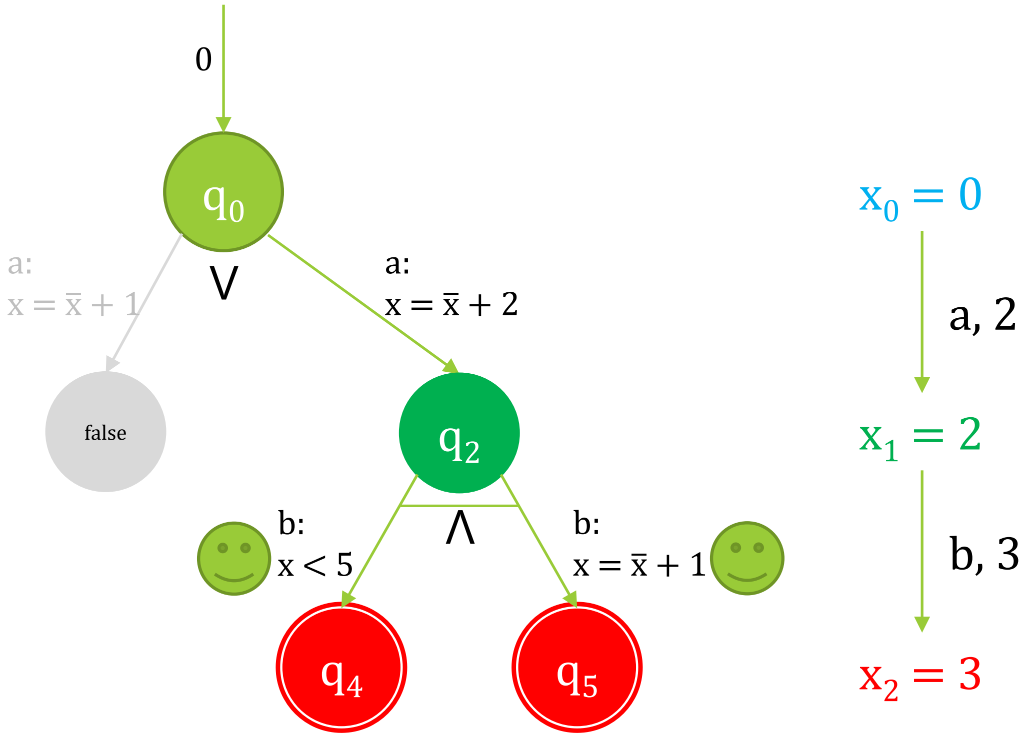
Data Words $w \in (\Sigma \times D^X)^*$

For example, the left alternating data automaton accepts the data word:
$(a, 2)\ (b, 3)$

Previous Data	**Current Data**
$x_1 = 2$	$x_2 = 3$

Alternating Data Automata

Accepting Word



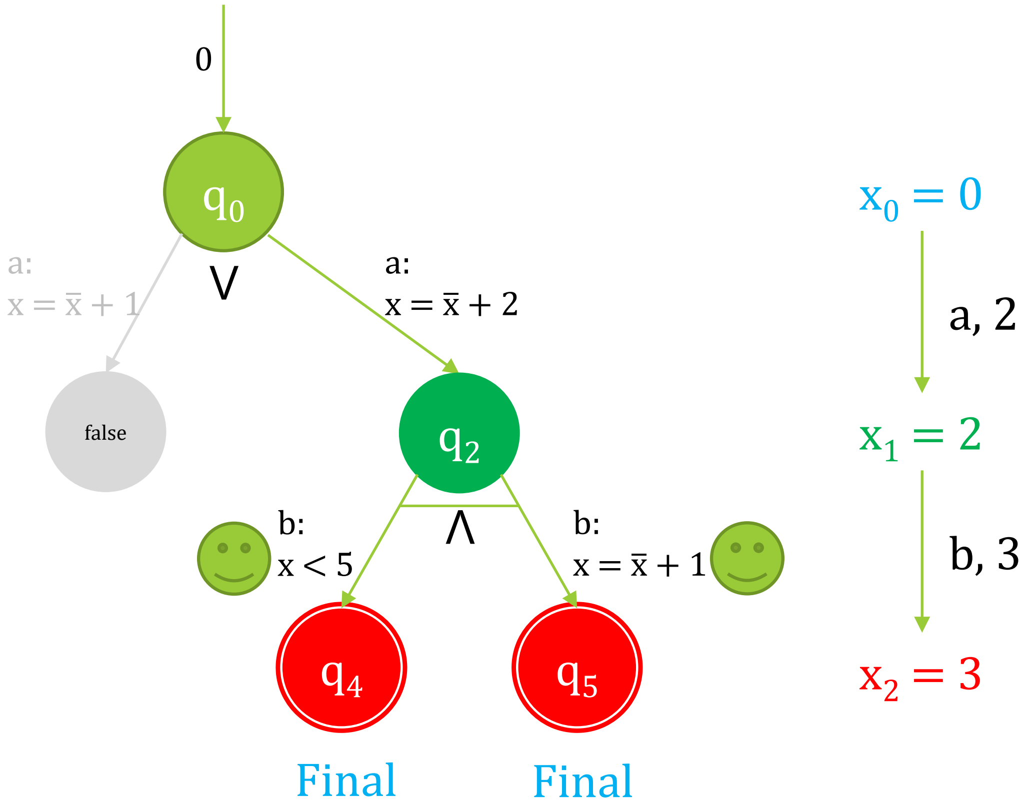
Data Words $w \in (\Sigma \times D^X)^*$

For example, the left alternating data automaton accepts the data word:
$(a, 2)\ (b, 3)$

Previous Data	Current Data
$x_1 = 2$	$x_2 = 3$

Alternating Data Automata

Accepting Word



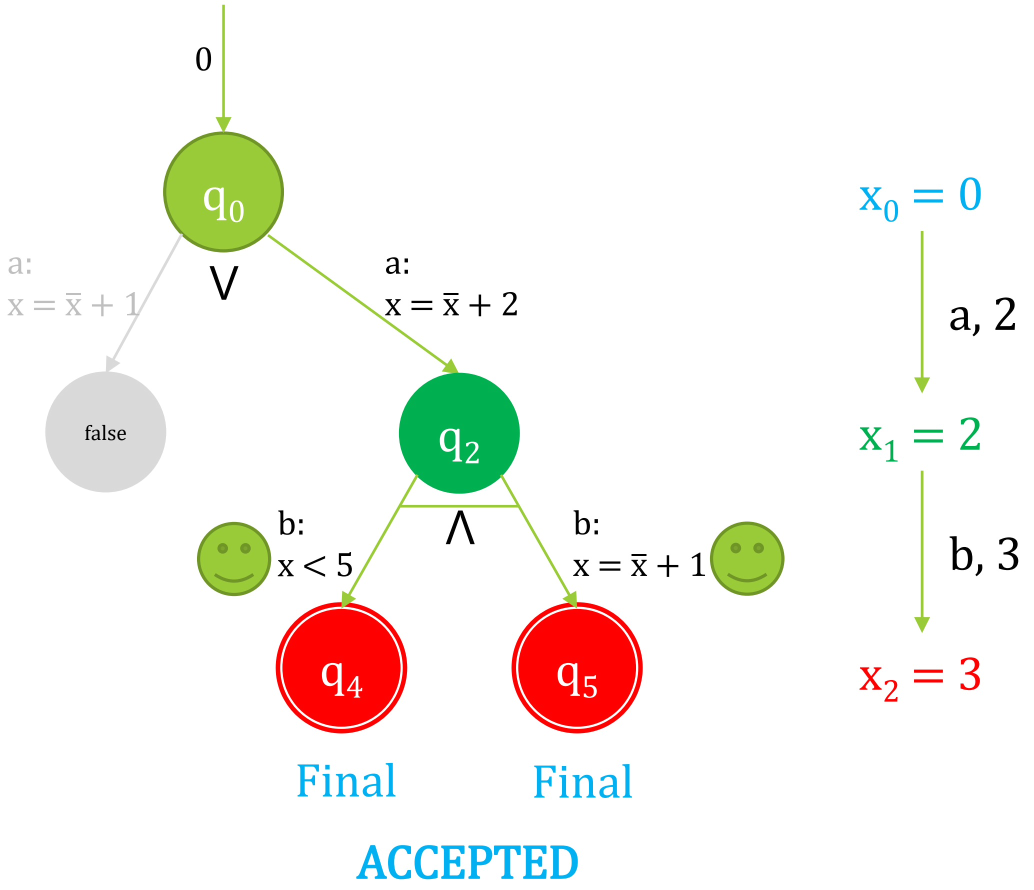
Data Words $w \in (\Sigma \times D^X)^*$

For example, the left alternating data automaton accepts the data word:
$(a, 2)\ (b, 3)$

Previous Data	**Current Data**
$x_1 = 2$	$x_2 = 3$

Alternating Data Automata

Accepting Word



Data Words $w \in (\Sigma \times D^X)^*$

For example, the left alternating data automaton accepts the data word:
$(a, 2)\ (b, 3)$

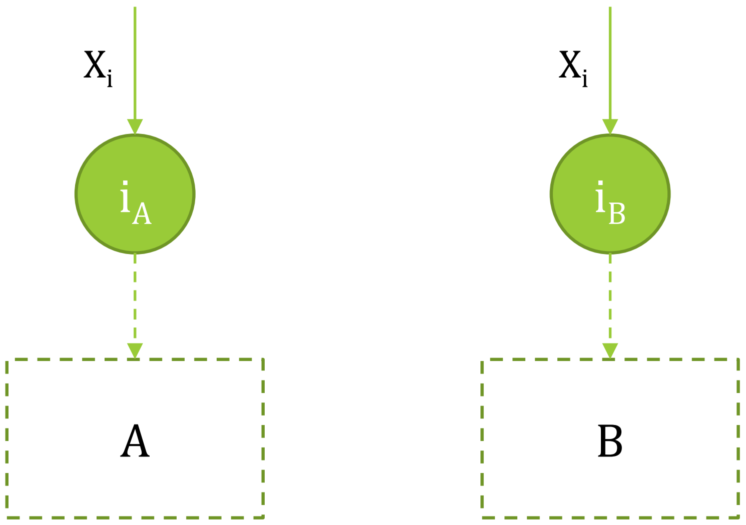
Previous Data	Current Data
$x_1 = 2$	$x_2 = 3$

Alternating Data Automata

Intersection

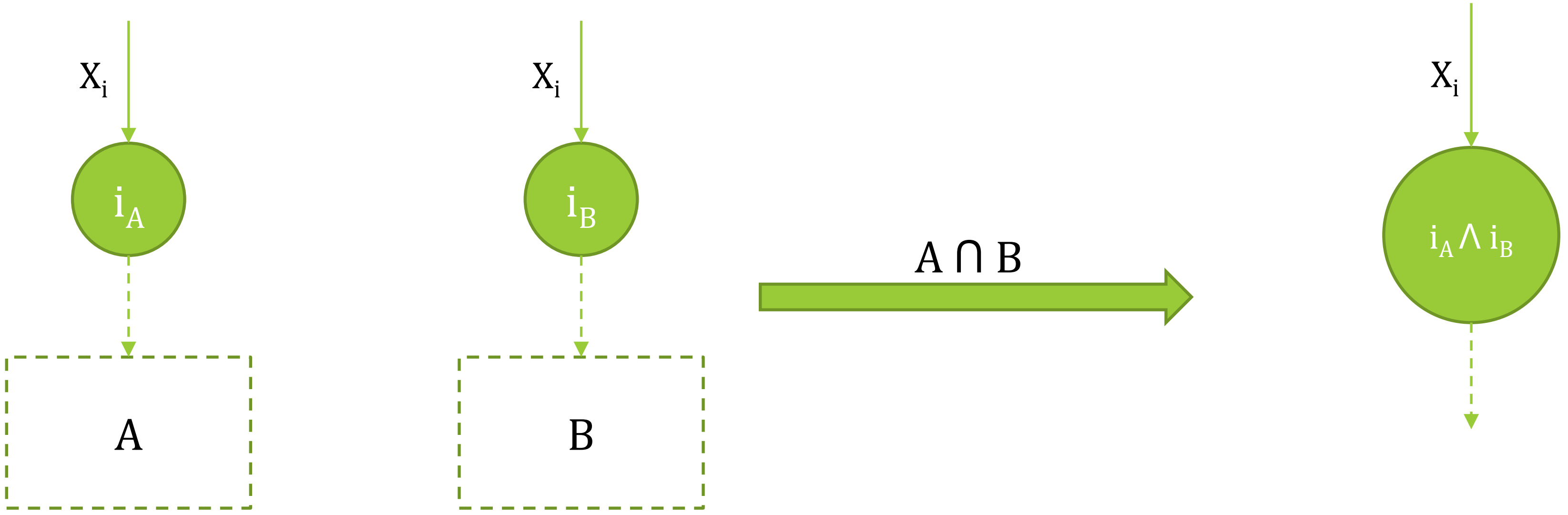
Alternating Data Automata

Intersection



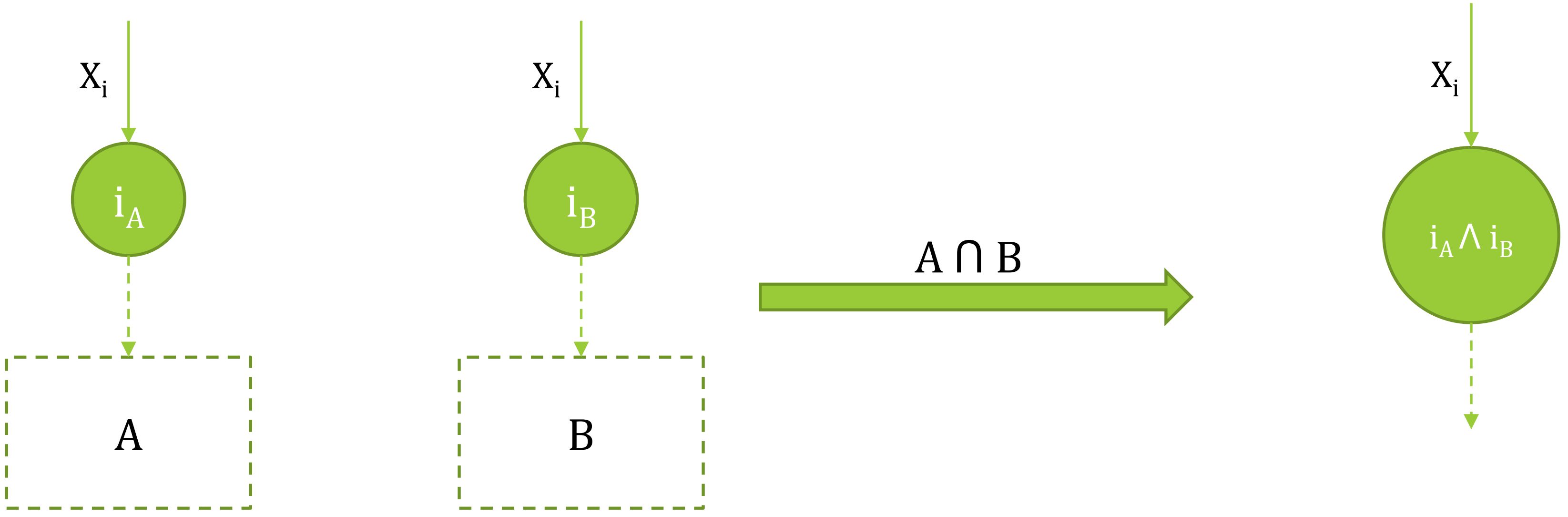
Alternating Data Automata

Intersection



Alternating Data Automata

Intersection



Linear Time

Alternating Data Automata

Complementation

Alternating Data Automata
Complementation

$A = < X, Q, \iota, F, \Delta >$

Alternating Data Automata

Complementation

$A = < X, Q, \iota, F, \Delta >$

$\overline{A} = < X, Q, \iota, Q - F, \Delta' >$

Alternating Data Automata

Complementation

$A = < X, Q, \iota, F, \Delta >$

$\overline{A} = < X, Q, \iota, Q - F, \Delta' >$

Alternating Data Automata

Complementation

$A = < X, Q, \iota, F, \Delta >$

$\overline{A} = < X, Q, \iota, Q - F, \Delta' >$

$$q_1 \wedge x = \bar{x} + 1 \vee q_2 \wedge x = \bar{x} + 2$$

$$\Downarrow$$

$$(q_1 \vee x \neq \bar{x} + 1) \wedge (q_2 \vee x \neq \bar{x} + 2)$$

Alternating Data Automata

Complementation

$A = < X, Q, \iota, F, \Delta >$

$\overline{A} = < X, Q, \iota, Q - F, \Delta' >$

$$q_1 \land x = \bar{x} + 1 \lor q_2 \land x = \bar{x} + 2$$

$$\Downarrow$$

$$(q_1 \lor x \neq \bar{x} + 1) \land (q_2 \lor x \neq \bar{x} + 2)$$

Alternating Data Automata
Complementation

$A = < X, Q, \iota, F, \Delta >$

$\overline{A} = < X, Q, \iota, Q - F, \Delta' >$

$$q_1 \wedge x = \bar{x} + 1 \vee q_2 \wedge x = \bar{x} + 2$$

$$\Downarrow$$

$$(q_1 \vee x \neq \bar{x} + 1) \wedge (q_2 \vee x \neq \bar{x} + 2)$$

Alternating Data Automata

Complementation

$A = \langle X, Q, \iota, F, \Delta \rangle$

$\overline{A} = \langle X, Q, \iota, Q - F, \Delta' \rangle$

$$q_1 \wedge x = \bar{x} + 1 \vee q_2 \wedge x = \bar{x} + 2$$

$$\Downarrow$$

$$(q_1 \vee x \neq \bar{x} + 1) \wedge (q_2 \vee x \neq \bar{x} + 2)$$

Alternating Data Automata

Complementation

$A = < X, Q, \iota, F, \Delta >$

$\overline{A} = < X, Q, \iota, Q - F, \Delta' >$

$$q_1 \wedge x = \bar{x} + 1 \vee q_2 \wedge x = \bar{x} + 2$$
$$\Downarrow$$
$$(q_1 \vee x \neq \bar{x} + 1) \wedge (q_2 \vee x \neq \bar{x} + 2)$$

Δ

	a	b
q0	$q_1 \wedge x = \bar{x} + 1 \vee q_2 \wedge x = \bar{x} + 2$	false
q1	false	$q_3 \wedge x < 3$
q2	false	$q_4 \wedge x < 5 \wedge q_5 \wedge x = \bar{x} + 1$
q3	false	false
q4	false	false
q5	false	false

Δ'

	a	b
q0	$(q_1 \vee x \neq \bar{x} + 1) \wedge (q_2 \vee x \neq \bar{x} + 2)$	true
q1	true	$q_3 \vee x \geq 3$
q2	true	$q_4 \vee x \geq 5 \vee q_5 \vee x \neq \bar{x} + 1$
q3	true	true
q4	true	true
q5	true	true

Alternating Data Automata

Complementation

$A = < X, Q, \iota, F, \Delta >$

$\overline{A} = < X, Q, \iota, Q - F, \Delta' >$

$$q_1 \land x = \bar{x} + 1 \lor q_2 \land x = \bar{x} + 2$$

$$\Downarrow$$

$$(q_1 \lor x \neq \bar{x} + 1) \land (q_2 \lor x \neq \bar{x} + 2)$$

Linear Time
(all variables are visible)

Δ

	a	b
q0	$q_1 \land x = \bar{x} + 1 \lor q_2 \land x = \bar{x} + 2$	false
q1	false	$q_3 \land x < 3$
q2	false	$q_4 \land x < 5 \land q_5 \land x = \bar{x} + 1$
q3	false	false
q4	false	false
q5	false	false

Δ'

	a	b
q0	$(q_1 \lor x \neq \bar{x} + 1) \land (q_2 \lor x \neq \bar{x} + 2)$	true
q1	true	$q_3 \lor x \geq 3$
q2	true	$q_4 \lor x \geq 5 \lor q_5 \lor x \neq \bar{x} + 1$
q3	true	true
q4	true	true
q5	true	true

Emptiness of Alternating Data Automata

Emptiness of Alternating Data Automata

Interpolation

Emptiness of Alternating Data Automata
Interpolation

Emptiness Problem

- Undecidable

Emptiness of Alternating Data Automata

Interpolation

Emptiness Problem

- Undecidable

Semi-Algorithms (with Anti-Chains) Based on Abstraction(Over-Approximation) and Refinement

- Lazy Predicate Abstraction
- Impact

Emptiness of Alternating Data Automata
Interpolation

Emptiness Problem
- Undecidable

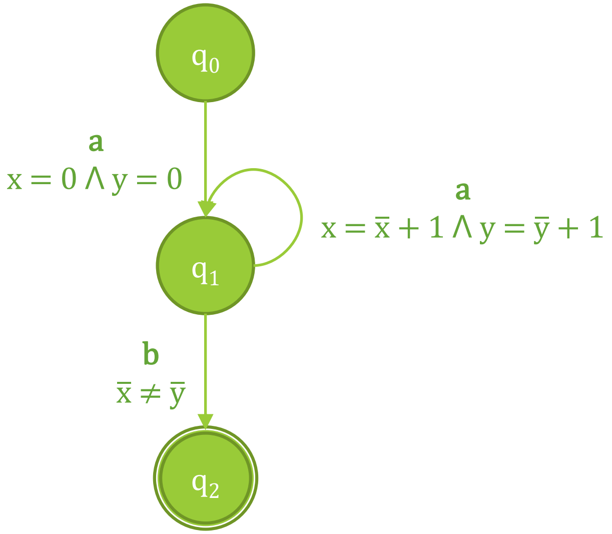
Semi-Algorithms (with Anti-Chains) Based on Abstraction(Over-Approximation) and Refinement
- Lazy Predicate Abstraction
- Impact

Emptiness is proved with abstraction:
- The automaton is surely empty

Counter-example (that proves the non-emptiness) is found with abstraction:
- The counter-example can be spurious
- Need to refine (compute the interpolants)

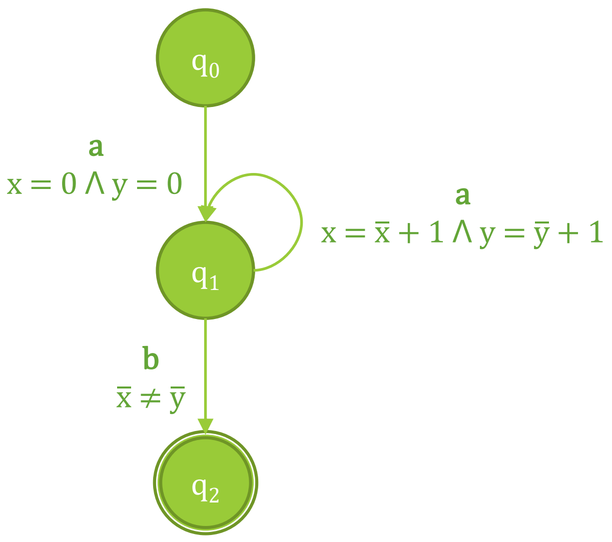
Emptiness of Alternating Data Automata

Interpolation



Emptiness of Alternating Data Automata

Interpolation



$$q_0$$

$$\downarrow a$$

$$q_1 \wedge x = 0 \wedge y = 0 \xrightarrow{b} \bot$$

$$\downarrow a$$

$$q_1 \wedge x = 1 \wedge y = 1 \xrightarrow{b} \bot$$

$$\downarrow a$$

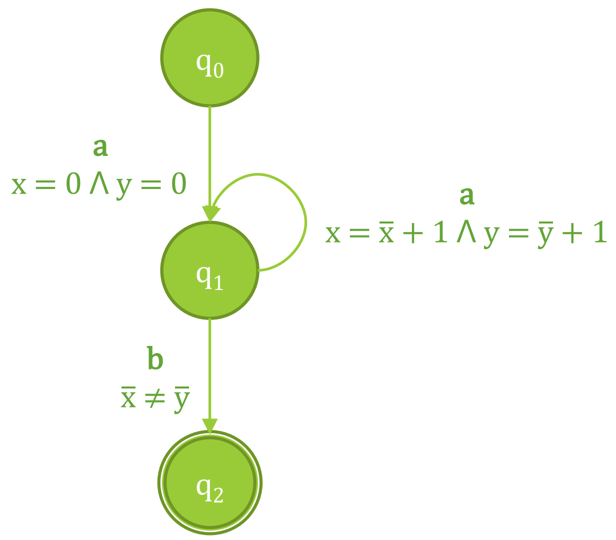
$$q_1 \wedge x = 2 \wedge y = 2 \xrightarrow{b} \bot$$

$$\downarrow a$$

…

Emptiness of Alternating Data Automata

Interpolation



$$q_0$$

$$\downarrow a$$

$$q_1 \wedge x = 0 \wedge y = 0 \xrightarrow{b} \perp$$

$$\downarrow a$$

$$q_1 \wedge x = 1 \wedge y = 1 \xrightarrow{b} \perp$$

$$\downarrow a$$

$$q_1 \wedge x = 2 \wedge y = 2 \xrightarrow{b} \perp$$

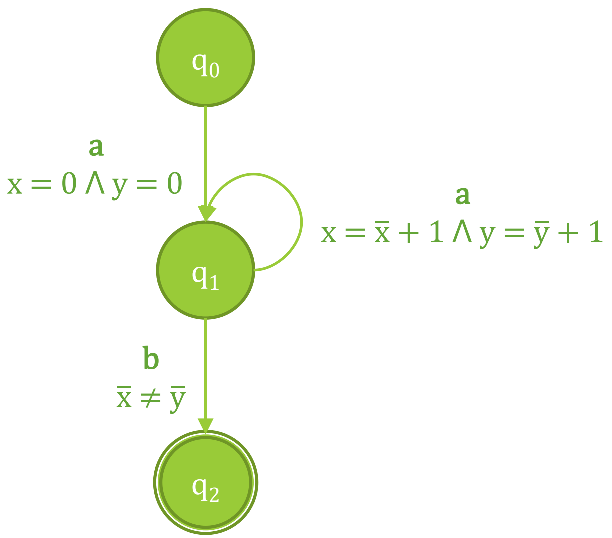
$$\downarrow a$$

…

Will never terminate…

Emptiness of Alternating Data Automata

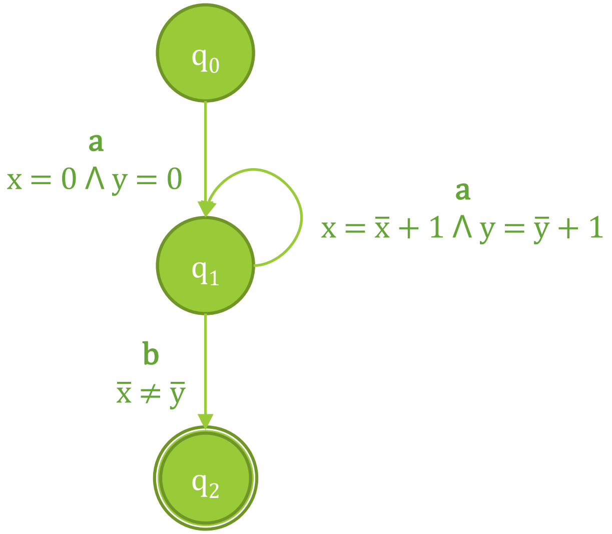
Interpolation



q_0

Emptiness of Alternating Data Automata

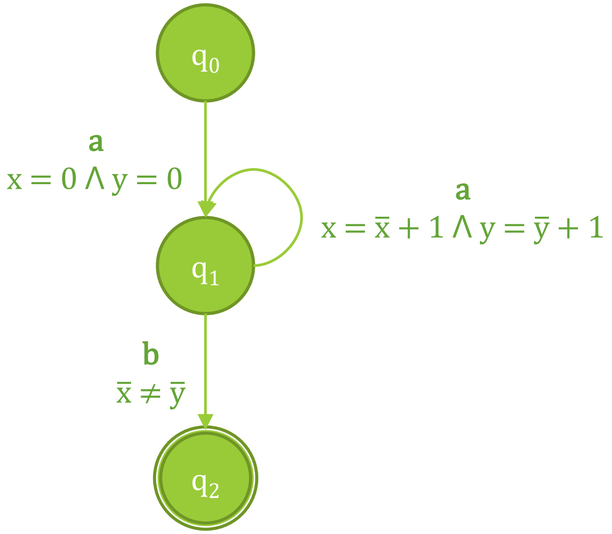
Interpolation



$$q_0 \xrightarrow{a} q_1 \wedge x = 0 \wedge y = 0 \xrightarrow{b} \perp$$

Emptiness of Alternating Data Automata

Interpolation

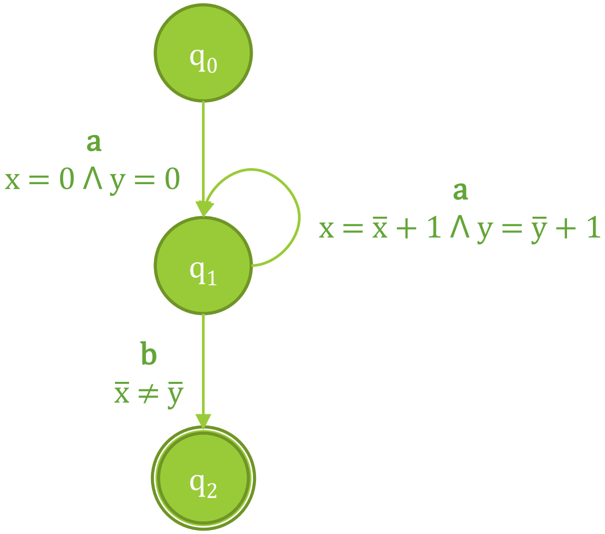


$$q_0 \xrightarrow{a} q_1 \wedge x = 0 \wedge y = 0 \xrightarrow{b} \bot$$

$$(q_1 \wedge x = 0 \wedge y = 0) \wedge (x \neq y) \rightarrow \bot$$

Emptiness of Alternating Data Automata

Interpolation

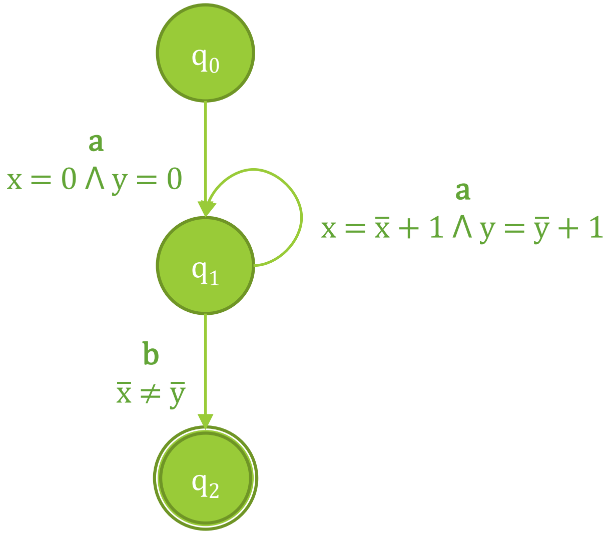


$$q_0 \xrightarrow{a} q_1 \wedge x = 0 \wedge y = 0 \xrightarrow{b} \perp$$

$(q_1 \wedge x = 0 \wedge y = 0) \wedge (x \neq y) \rightarrow \perp$
get an interpolant: $q_1 \wedge x = y$
which is an over-approximation of $q_1 \wedge x = 0 \wedge y = 0$
we have:
$(q_1 \wedge x = 0 \wedge y = 0) \rightarrow (q_1 \wedge x = y)$
$(q_1 \wedge x = y) \wedge (x \neq y) \rightarrow \perp$

Emptiness of Alternating Data Automata

Interpolation



$$
\begin{array}{c}
q_0 \\
\big\downarrow{\scriptstyle a} \\
\cancel{q_1 \wedge x = 0 \wedge y = 0} \xrightarrow{b} \bot \\
q_1 \wedge x = y
\end{array}
$$

$(q_1 \wedge x = 0 \wedge y = 0) \wedge (x \neq y) \rightarrow \bot$

get an interpolant: $q_1 \wedge x = y$

which is an over-approximation of $q_1 \wedge x = 0 \wedge y = 0$

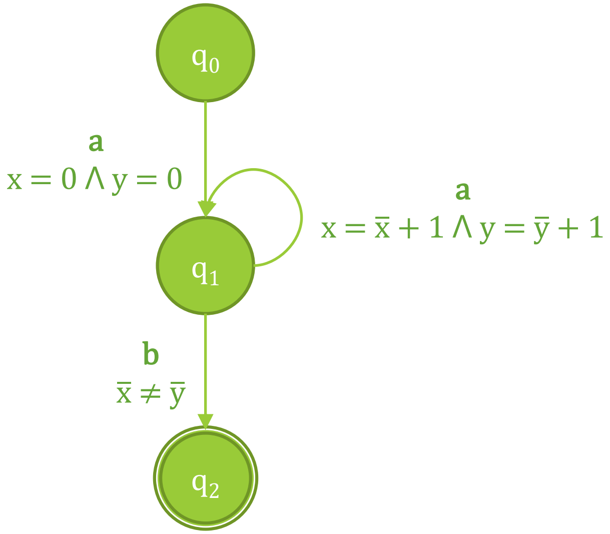
we have:

$(q_1 \wedge x = 0 \wedge y = 0) \rightarrow (q_1 \wedge x = y)$

$(q_1 \wedge x = y) \wedge (x \neq y) \rightarrow \bot$

Emptiness of Alternating Data Automata

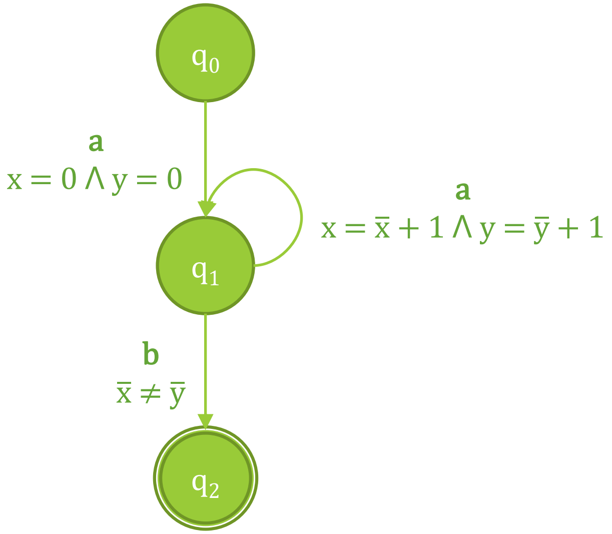
Interpolation



$$q_0 \xrightarrow{a} q_1 \land x = y \xrightarrow{b} \bot$$

Emptiness of Alternating Data Automata

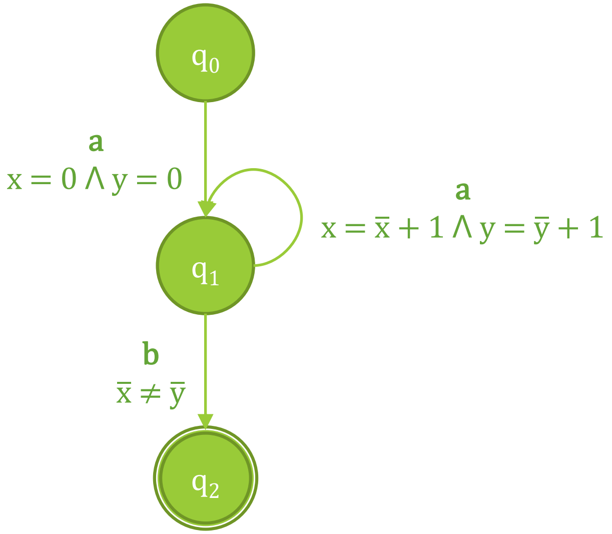
Interpolation

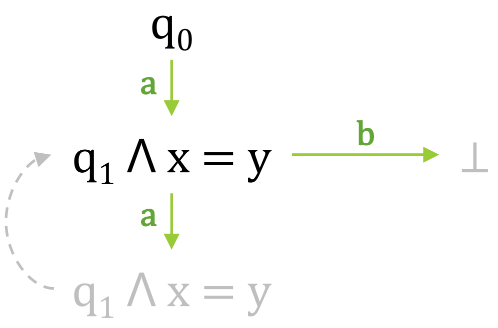


$$\begin{tikzcd}
q_0 \arrow[d, "a"'] & \\
q_1 \wedge x = y \arrow[r, "b"] \arrow[d, "a"'] & \bot \\
q_1 \wedge x = y &
\end{tikzcd}$$

Emptiness of Alternating Data Automata

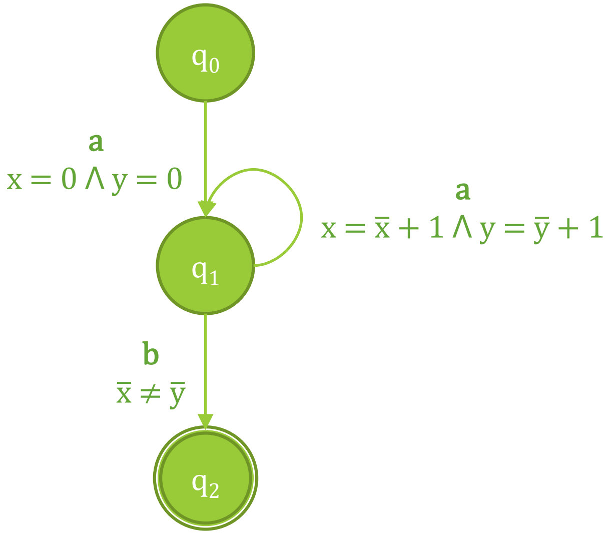
Interpolation

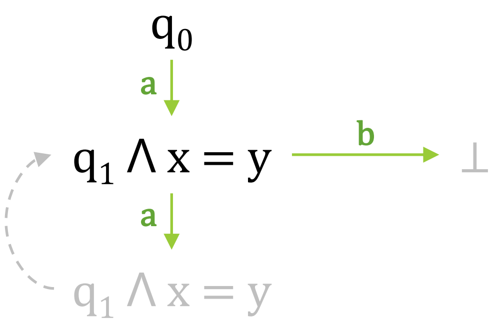




Emptiness of Alternating Data Automata

Interpolation





EMPTY

Emptiness of Alternating Data Automata

Interpolation

$\varphi_1 \land \varphi_2 \land \ldots \land \varphi_n \rightarrow \bot$

Emptiness of Alternating Data Automata

Interpolation

$\varphi_1 \land \varphi_2 \land \ldots \land \varphi_n \rightarrow \bot$

$\ldots$

$I_1 \quad I_2 \ldots I_{n-1}$

$\top \land \varphi_1 \rightarrow I_1$

$I_1 \land \varphi_2 \rightarrow I_2$

…

$I_{n-2} \land \varphi_{n-1} \rightarrow I_{n-1}$

$I_{n-1} \land \varphi_n \rightarrow \bot$

Emptiness of Alternating Data Automata

Interpolation

$\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n \rightarrow \bot$

$I_1 \quad I_2 \ldots I_{n-1}$

$\top \wedge \varphi_1 \rightarrow I_1$

$I_1 \wedge \varphi_2 \rightarrow I_2$

…

$I_{n-2} \wedge \varphi_{n-1} \rightarrow I_{n-1}$

$I_{n-1} \wedge \varphi_n \rightarrow \bot$

$FV(I_1) \subseteq FV(\varphi_1) \cap FV(\varphi_2)$

$FV(I_2) \subseteq FV(\varphi_2) \cap FV(\varphi_3)$

…

$FV(I_{n-1}) \subseteq FV(\varphi_{n-1}) \cap FV(\varphi_n)$

$\varphi_1(x_1) \wedge \varphi_2(x_1, x_2) \wedge \ldots \wedge \varphi_n(x_{n-1}, x_n) \rightarrow \bot$

$I_1(x_1) \quad I_2(x_2) \ldots I_{n-1}(x_{n-1})$

$\top \wedge \varphi_1(x_1) \rightarrow I_1(x_1)$

$I_1(x_1) \wedge \varphi_2(x_1, x_2) \rightarrow I_2(x_2)$

…

$I_{n-2}(x_{n-2}) \wedge \varphi_{n-1}(x_{n-2}, x_{n-1}) \rightarrow I_{n-1}(x_{n-1})$

$I_{n-1}(x_{n-1}) \wedge \varphi_n(x_{n-1}, x_n) \rightarrow \bot$

Emptiness of Alternating Data Automata

Acceptance

Emptiness of Alternating Data Automata
Acceptance

Given a Boolean formula φ, for any $u \in \Sigma^*$, we define the function of acceptance:

$\mathrm{ACC}(\varphi, u) = \mathrm{POST}(\varphi, u) \wedge (\boldsymbol{\bigwedge}_{q \in Q \setminus F}(q \rightarrow \bot))$

Emptiness of Alternating Data Automata

Acceptance

Given a Boolean formula φ, for any $u \in \Sigma^*$, we define the function of acceptance:

$\mathrm{ACC}(\varphi, u) = \mathrm{POST}(\varphi, u) \wedge (\boldsymbol{\bigwedge}_{q \in Q \setminus F}(q \rightarrow \bot))$

The automaton is not empty iff there exists $u \in \Sigma^*$ such that ACC(ι, u) is satisfiable.

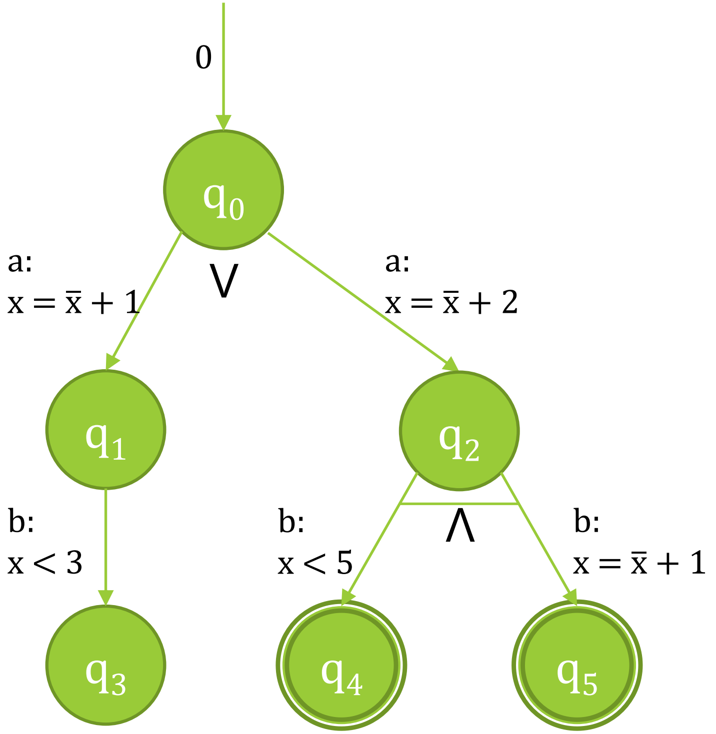
And we can have accepting words (counter-examples).

Emptiness of Alternating Data Automata

Acceptance

Given a Boolean formula φ, for any $u \in \Sigma^*$, we define the function of acceptance:

$\mathrm{ACC}(\varphi, u) = \mathrm{POST}(\varphi, u) \wedge (\bigwedge_{q \in Q \setminus F}(q \rightarrow \perp))$



Example:

$$\begin{aligned} & \mathrm{ACC}(q_0 \wedge x_0 = 0, ab) \\ = \ & (q_3 \wedge x_0 = 0 \wedge x_1 = 1 \wedge x_2 < 3) \vee (q_4 \wedge q_5 \wedge x_0 = 0 \wedge x_1 = 2 \wedge x_2 = 3) \\ & \wedge ((q_0 \rightarrow \perp) \wedge (q_1 \rightarrow \perp) \wedge (q_2 \rightarrow \perp) \wedge (q_3 \rightarrow \perp)) \end{aligned}$$

Satisfiable

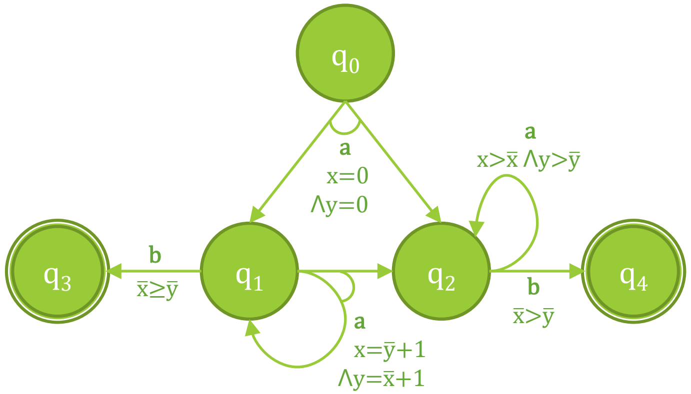
One Counter-Example(Accepting Word): $(a, 2)(b, 3)$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

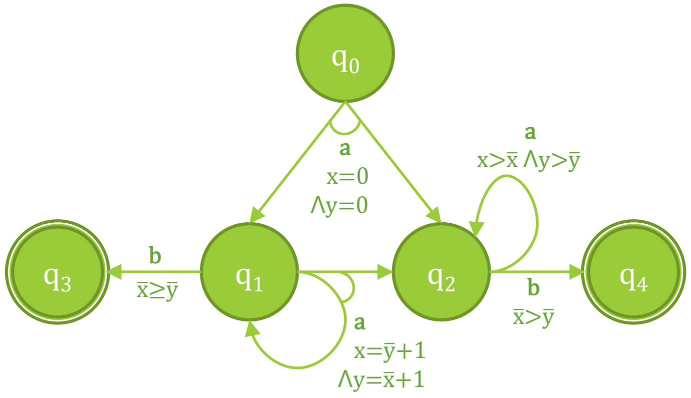
Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

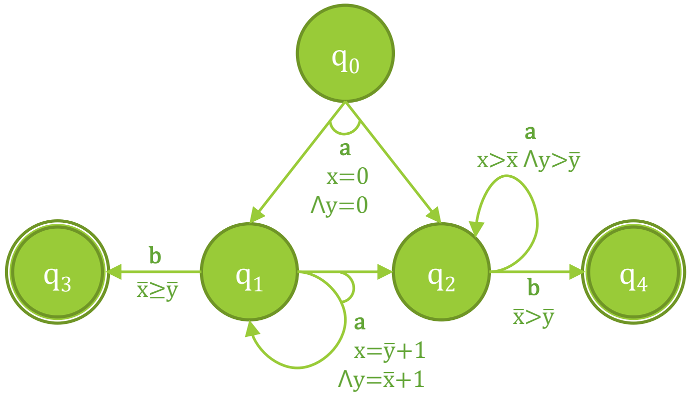


Π	⊥

Set of predicates that are used for excluding spurious counter-examples.

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



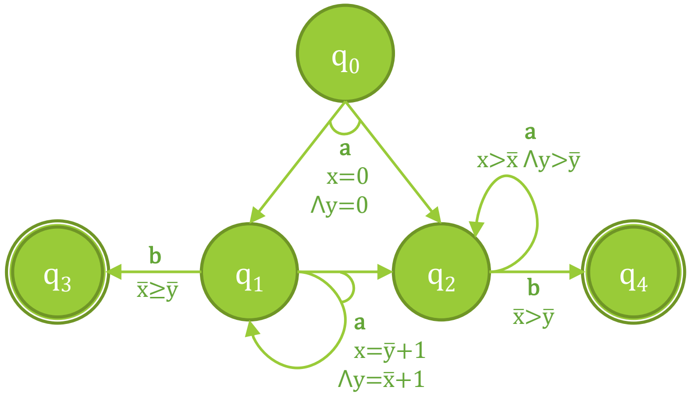
Π	⊥

Set of predicates that are used for excluding spurious counter-examples.

It is updated once we get new interpolants.

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



Π	$\perp$

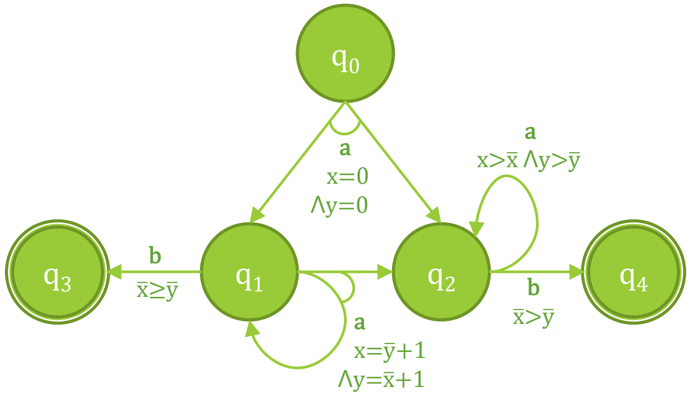
Set of predicates that are used for excluding spurious counter-examples.

It is updated once we get new interpolants.

Instead of working with concrete post image, at each step, we use the conjunction of all the elements from Π which can be implied by the concrete post image.

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



Π	$\bot$

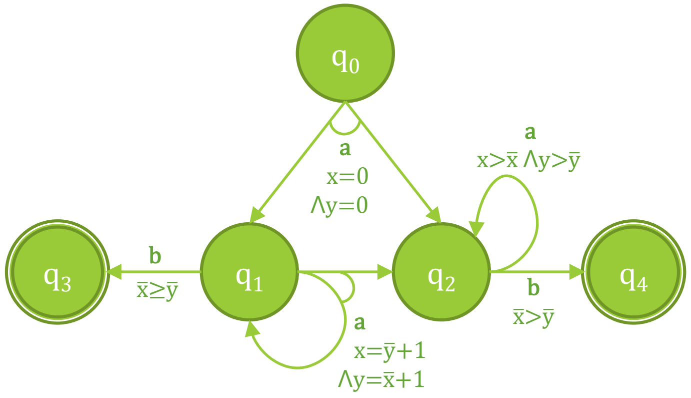
Set of predicates that are used for excluding spurious counter-examples.

It is updated once we get new interpolants.

Instead of working with concrete post image, at each step, we use the conjunction of all the elements from Π which can be implied by the concrete post image.
=> We work with the conjunction of the over-approximations of concrete post.

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

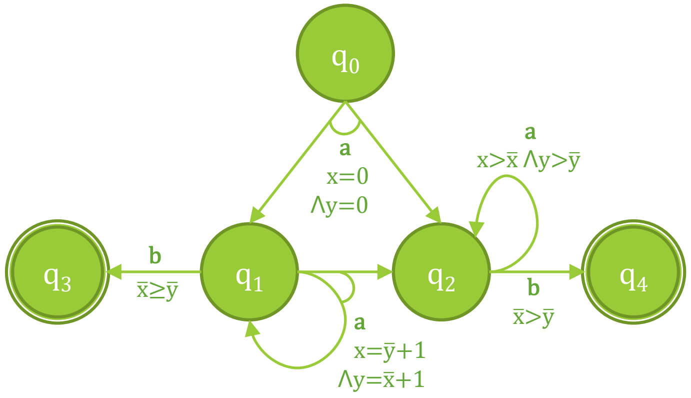


Π	⊥

q_0

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

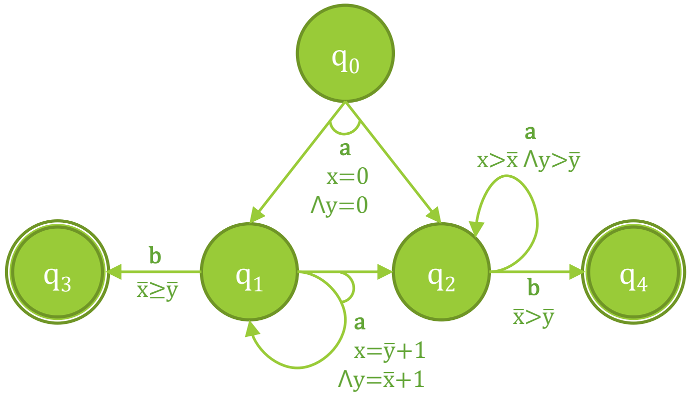


Π	⊥

$$q_0 \xrightarrow{a} \top$$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



Π	$\perp$, q_0, q_1

$q_0 \xrightarrow{a} \top$

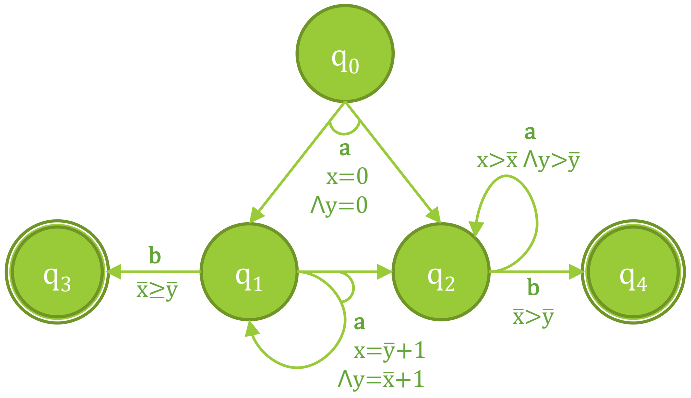
$\boxed{\text{pivot}}$

$ACC(q_0, a) = \perp$

Add Predicates: q_0, q_1

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

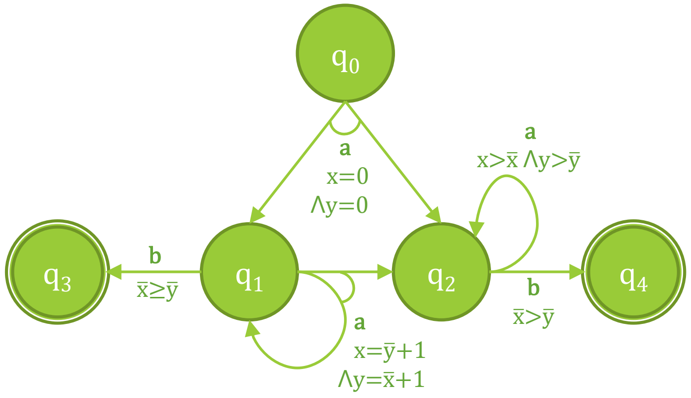


Π	⊥, q_0, q_1

q_0

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

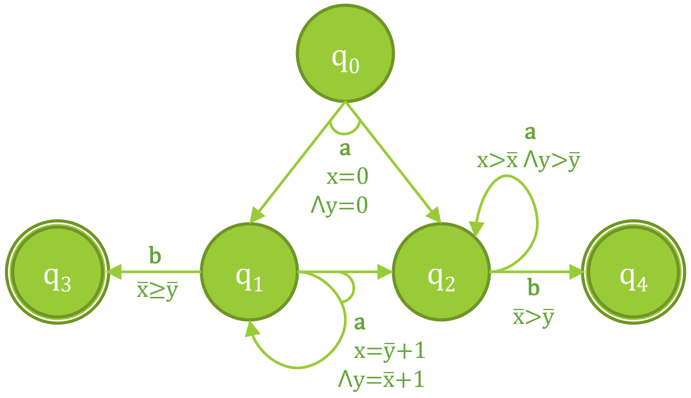


Π	$\perp, q_0, q_1$

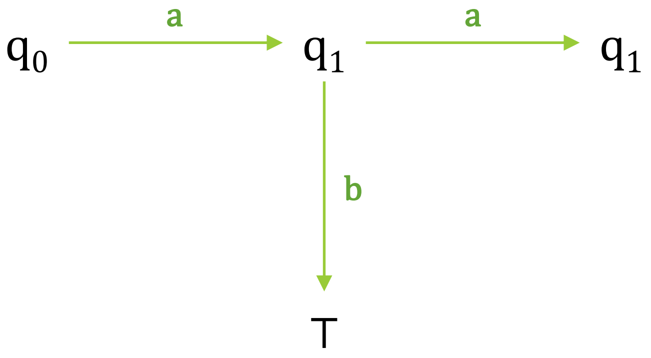
$$q_0 \xrightarrow{a} q_1$$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

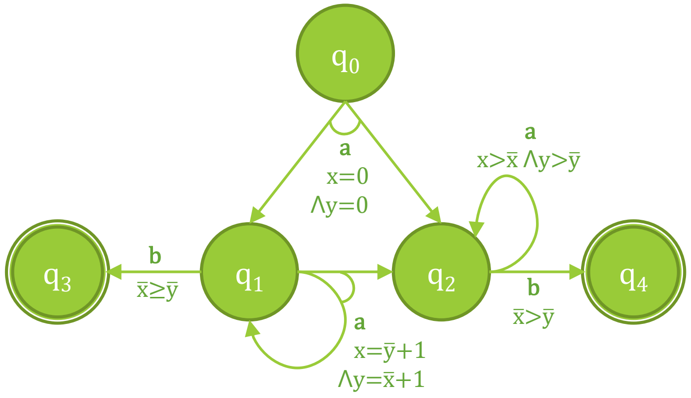


Π	$\perp, q_0, q_1$

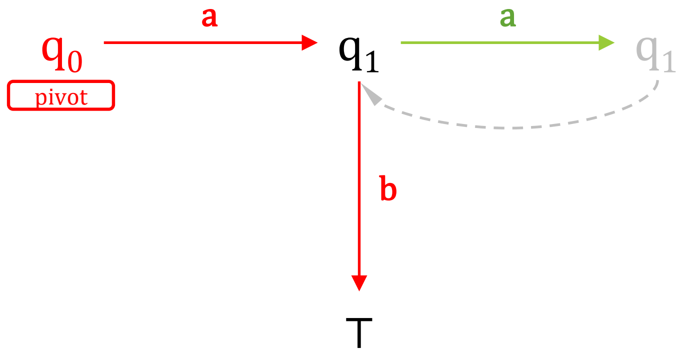


Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



Π	$\bot, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0$

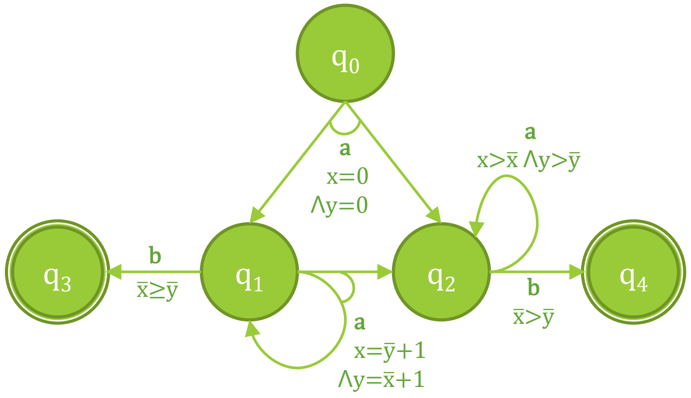


$ACC(q_0, ab) = \bot$

Add Predicate: $x \leq 0 \wedge q_2 \wedge y \geq 0$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

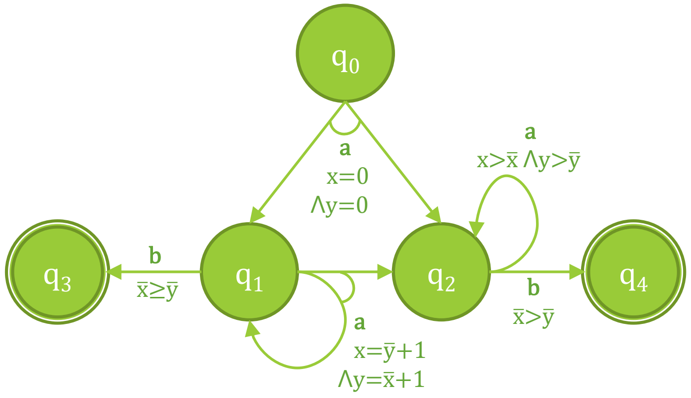


Π	$\bot, q_0, q_1, x \leq 0 \land q_2 \land y \geq 0$

q_0

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

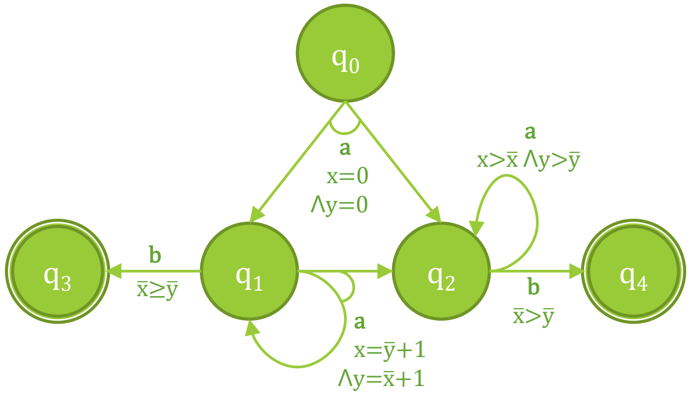


Π	$\perp, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0$

$$q_0 \xrightarrow{a} q_1 \wedge x \leq 0 \wedge q_2 \wedge y \geq 0$$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

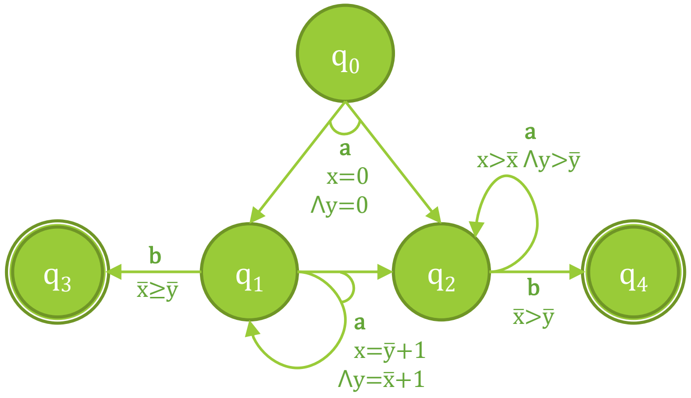


Π	$\perp, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0$

$$\begin{tikzcd}
q_0 \arrow[r, "a"] & q_1 \wedge x \leq 0 \wedge q_2 \wedge y \geq 0 \arrow[r, "b"] \arrow[d, "a"] & \perp \\
& q_1 &
\end{tikzcd}$$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

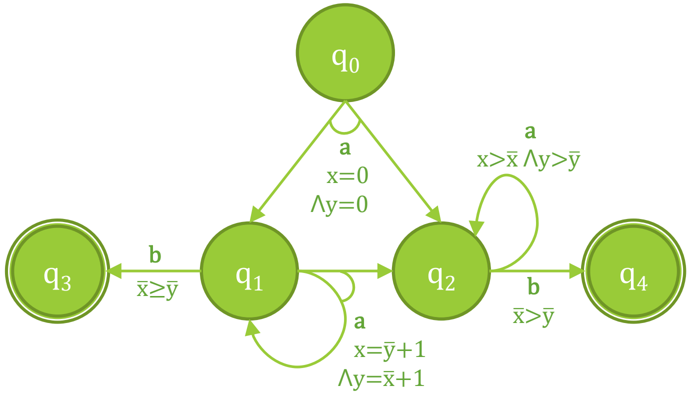


Π	$\bot, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0$

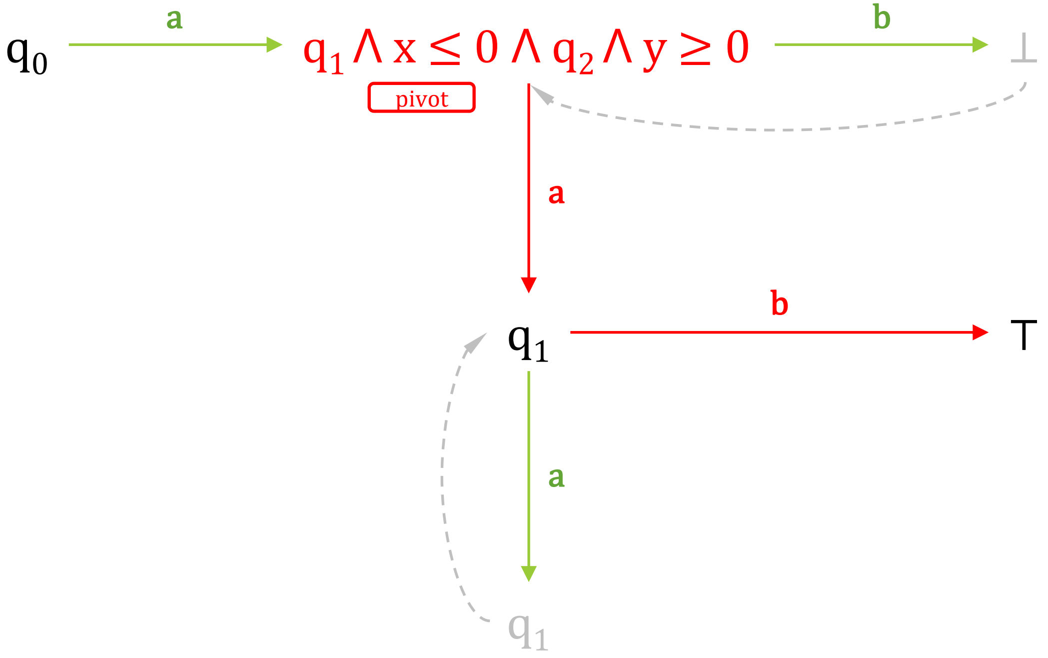
$$\begin{tikzcd}
q_0 \arrow[r, "a"] & q_1 \wedge x \leq 0 \wedge q_2 \wedge y \geq 0 \arrow[r, "b"] \arrow[d, "a"] & \bot \arrow[l, bend left, dashed] \\
& q_1 \arrow[r, "b"] \arrow[d, "a"] & \top \\
& q_1 &
\end{tikzcd}$$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



Π	$\perp, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0, q_1 \wedge q_2, y > x - 1 \wedge q_2$

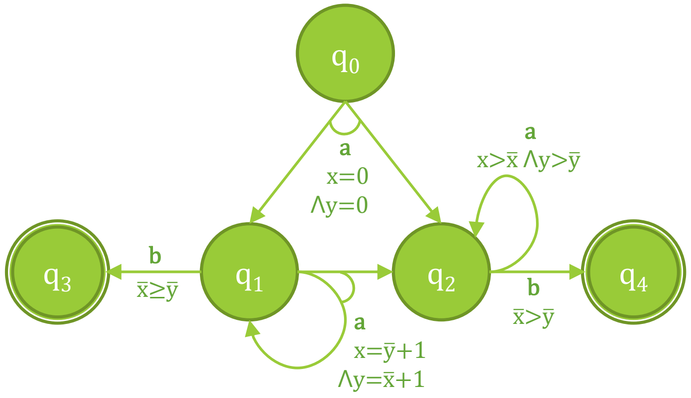


$ACC(q_1 \wedge x \leq 0 \wedge q_2 \wedge y \geq 0, ab) = \perp$

Add Predicates: $q_1 \wedge q_2, y > x - 1 \wedge q_2$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

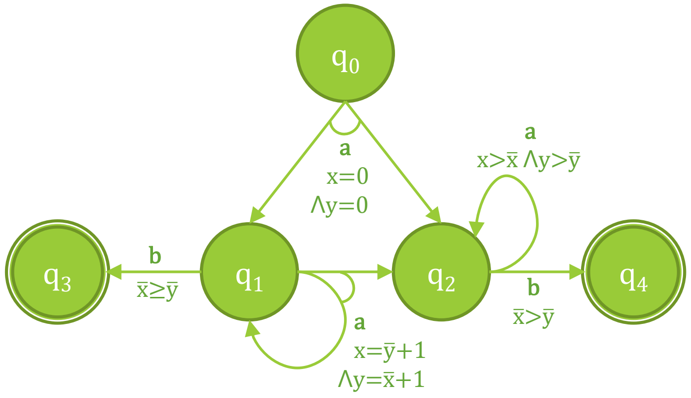


Π	$\perp, q_0, q_1, x \leq 0 \land q_2 \land y \geq 0, q_1 \land q_2, y > x - 1 \land q_2$

$q_0 \xrightarrow{a} q_1 \land x \leq 0 \land q_2 \land y \geq 0$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

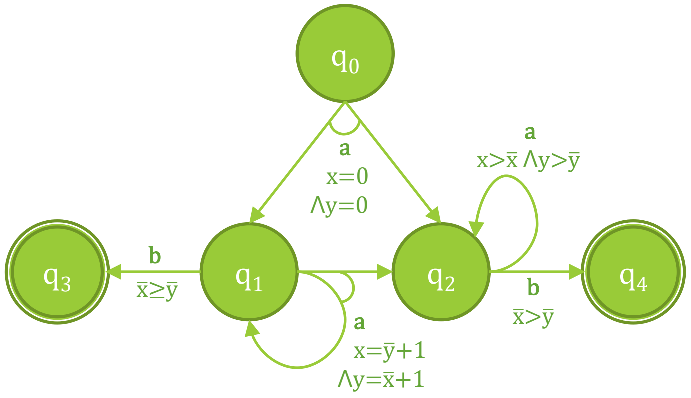


Π	$\bot, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0, q_1 \wedge q_2, y > x - 1 \wedge q_2$

$$\begin{tikzcd} q_0 \arrow[r, "a"] & q_1 \wedge x \leq 0 \wedge q_2 \wedge y \geq 0 \arrow[r, "b"] \arrow[d, "a"] & \bot \\ & q_1 \wedge y > x - 1 \wedge q_2 & \end{tikzcd}$$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

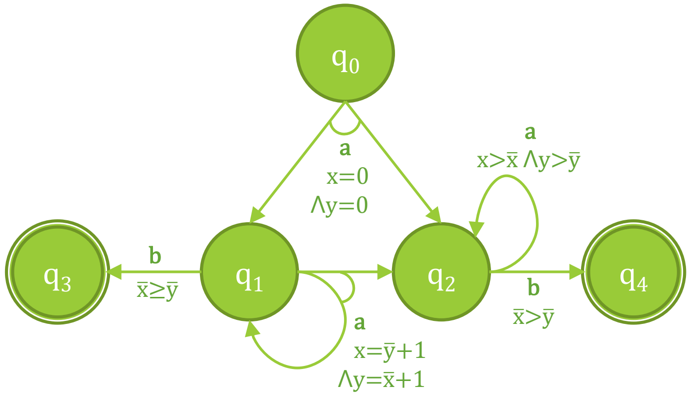


Π	$\perp, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0, q_1 \wedge q_2, y > x - 1 \wedge q_2$

$$\begin{tikzcd}
q_0 \arrow[r, "a"] & q_1 \wedge x \leq 0 \wedge q_2 \wedge y \geq 0 \arrow[r, "b"] \arrow[d, "a"] & \perp \arrow[l, dashed, bend left] \\
& q_1 \wedge y > x - 1 \wedge q_2 \arrow[r, "b"] \arrow[d, "a"] & \perp \\
& q_1 \wedge y > x - 1 \wedge q_2 &
\end{tikzcd}$$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



Π	$\perp, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0, q_1 \wedge q_2, y > x - 1 \wedge q_2$

$$q_0 \xrightarrow{a} q_1 \wedge x \leq 0 \wedge q_2 \wedge y \geq 0 \xrightarrow{b} \perp$$

$$\downarrow a$$

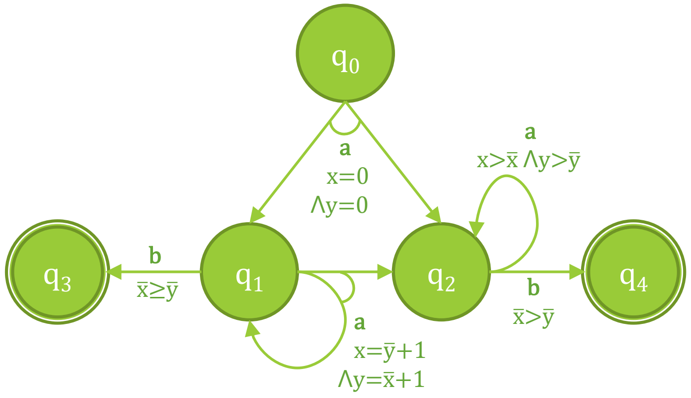
$$q_1 \wedge y > x - 1 \wedge q_2 \xrightarrow{b} \perp$$

$$\downarrow a$$

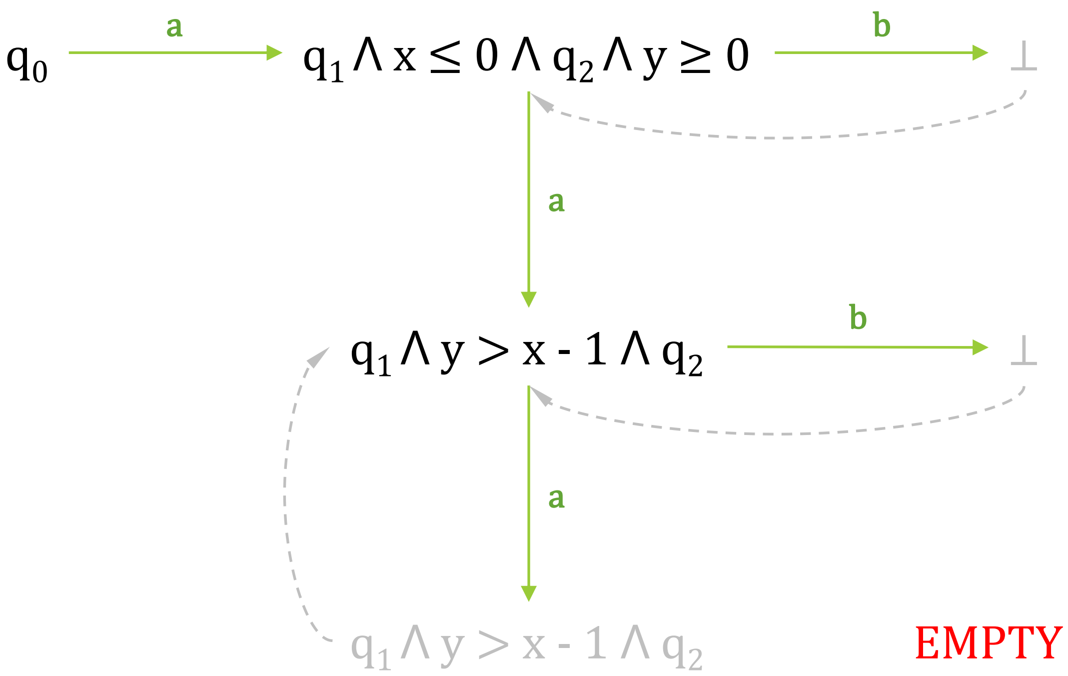
$$q_1 \wedge y > x - 1 \wedge q_2$$

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction



Π	$\perp, q_0, q_1, x \leq 0 \wedge q_2 \wedge y \geq 0, q_1 \wedge q_2, y > x - 1 \wedge q_2$



Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

<u>Problem</u>

After refining the set of predicates, we need to restart from pivot, then reconstruct the tree.

Emptiness of Alternating Data Automata

Lazy Predicate Abstraction

Problem

After refining the set of predicates, we need to restart from pivot, then reconstruct the tree.

HEAVY

Emptiness of Alternating Data Automata
Impact

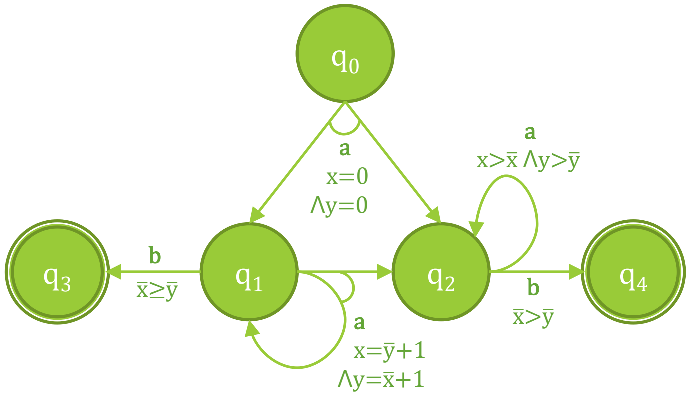
Problem

~~After refining the set of predicates, we need to restart from pivot, then reconstruct the tree.~~

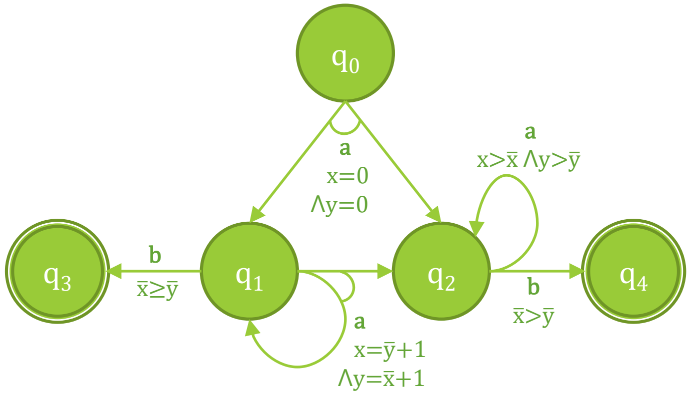
Do in-place refinement by strengthening the node labels.

Emptiness of Alternating Data Automata

Impact



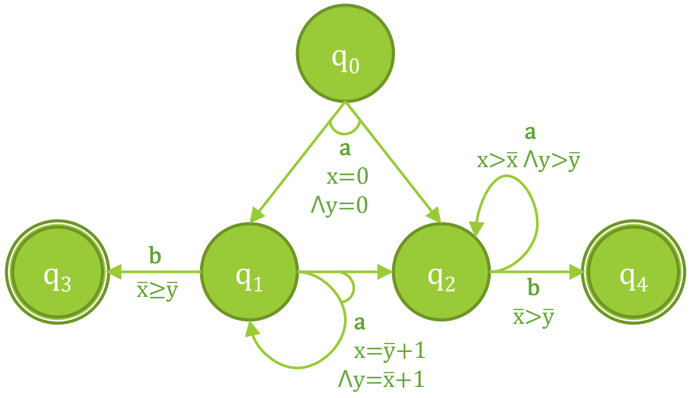
Emptiness of Alternating Data Automata
Impact



q_0

Emptiness of Alternating Data Automata

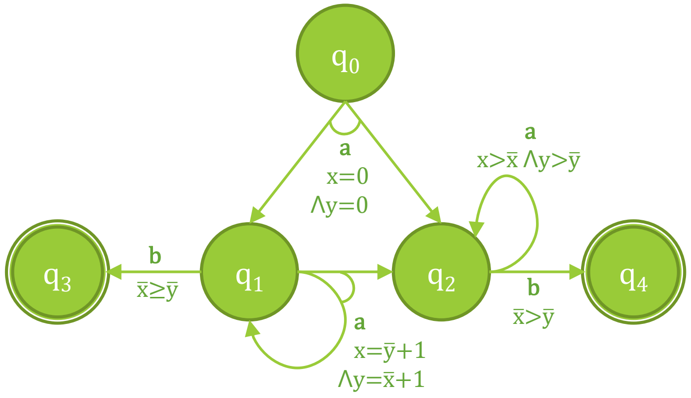
Impact



$q_0 \xrightarrow{a} \top$

Emptiness of Alternating Data Automata

Impact



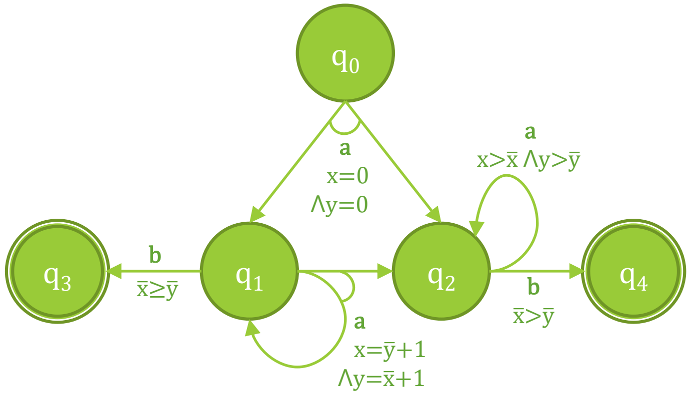
$q_0 \xrightarrow{a} \cancel{\top}\ \boxed{q_1}$

$ACC(q_0, a) = \bot$

Refine

Emptiness of Alternating Data Automata

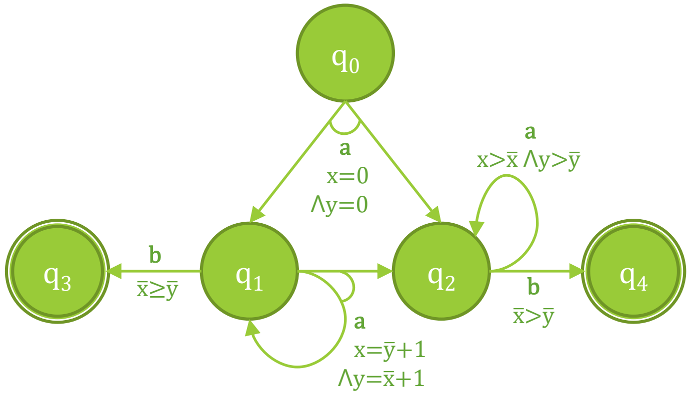
Impact

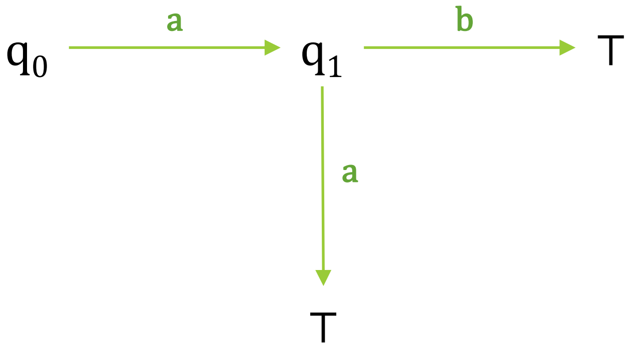


$$q_0 \xrightarrow{a} q_1$$

Emptiness of Alternating Data Automata

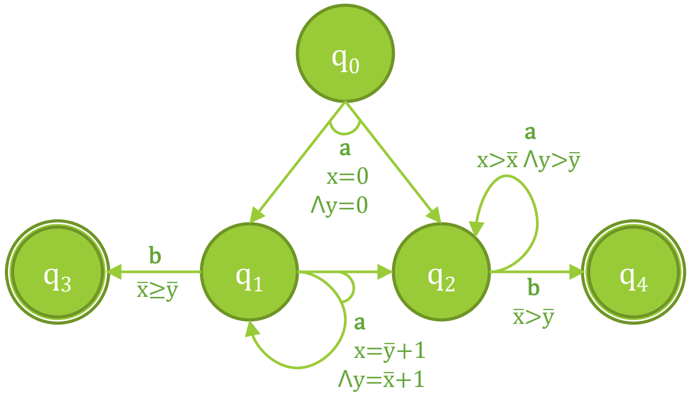
Impact

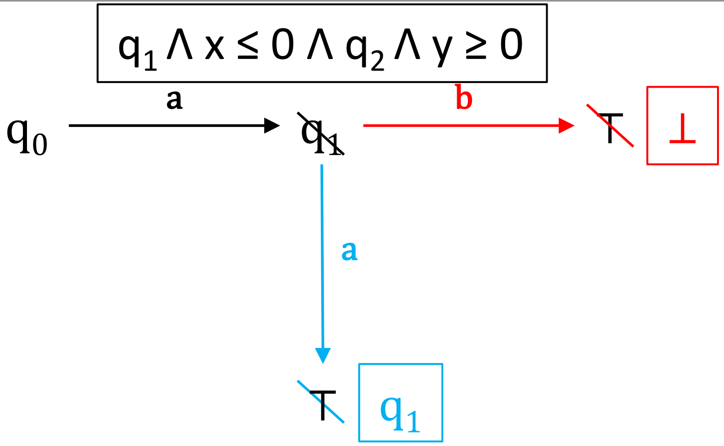




Emptiness of Alternating Data Automata

Impact





$ACC(q_0, aa) = \bot$

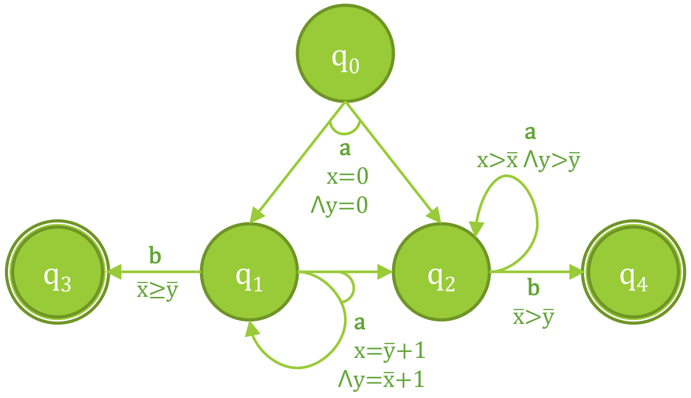
Refine

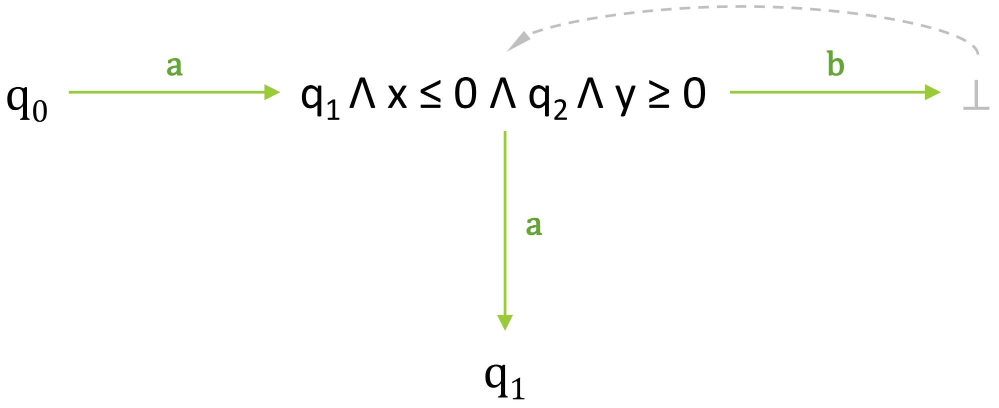
$ACC(q_0, ab) = \bot$

Refine

Emptiness of Alternating Data Automata

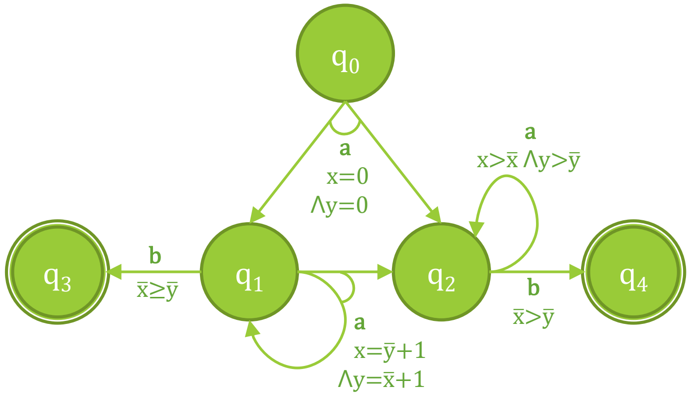
Impact

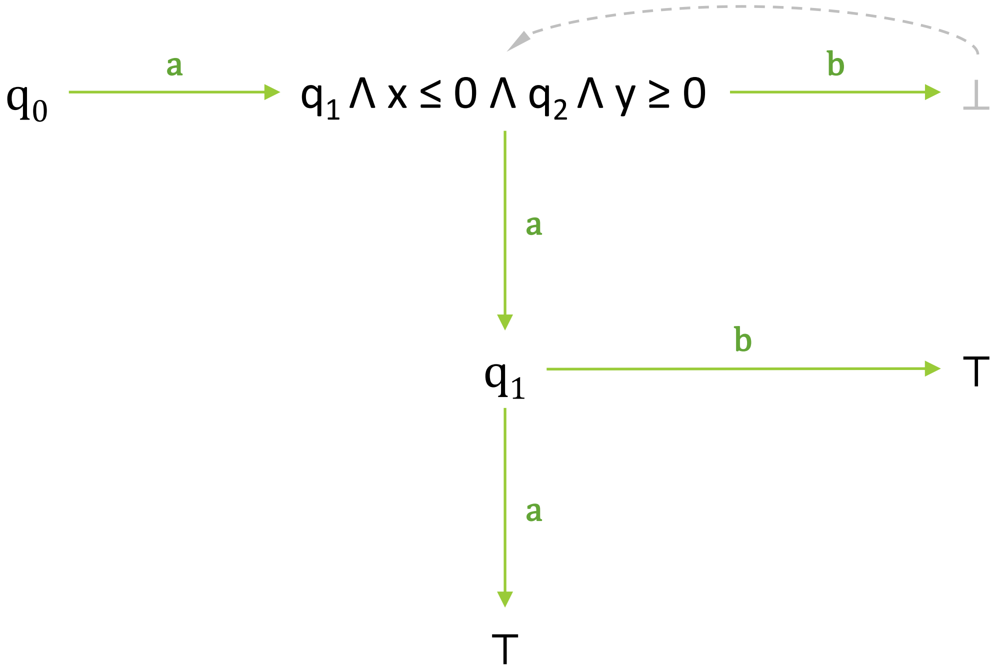




Emptiness of Alternating Data Automata

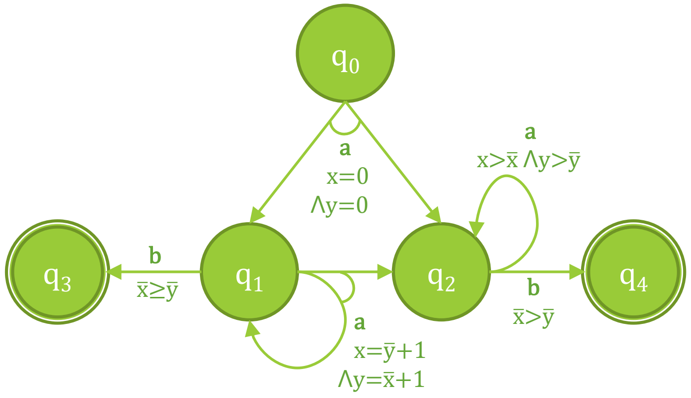
Impact

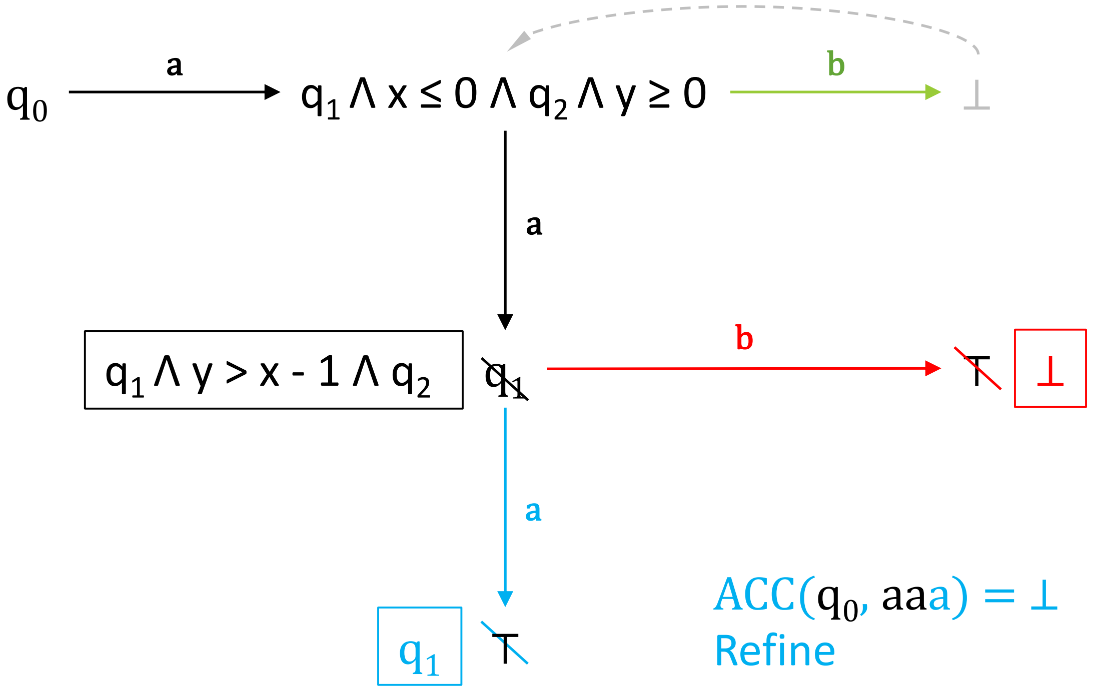




Emptiness of Alternating Data Automata

Impact





$\mathrm{ACC}(q_0, aaa) = \perp$

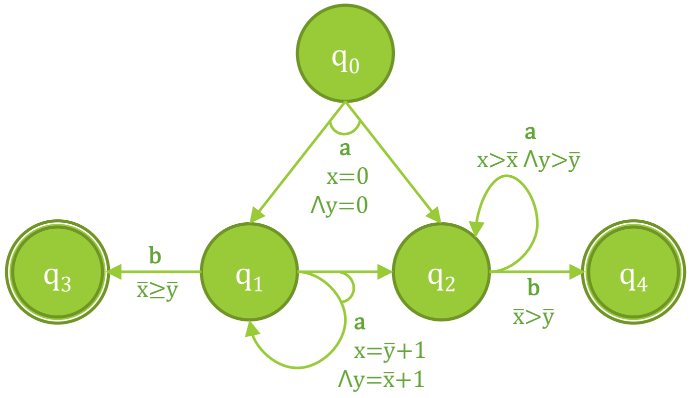
Refine

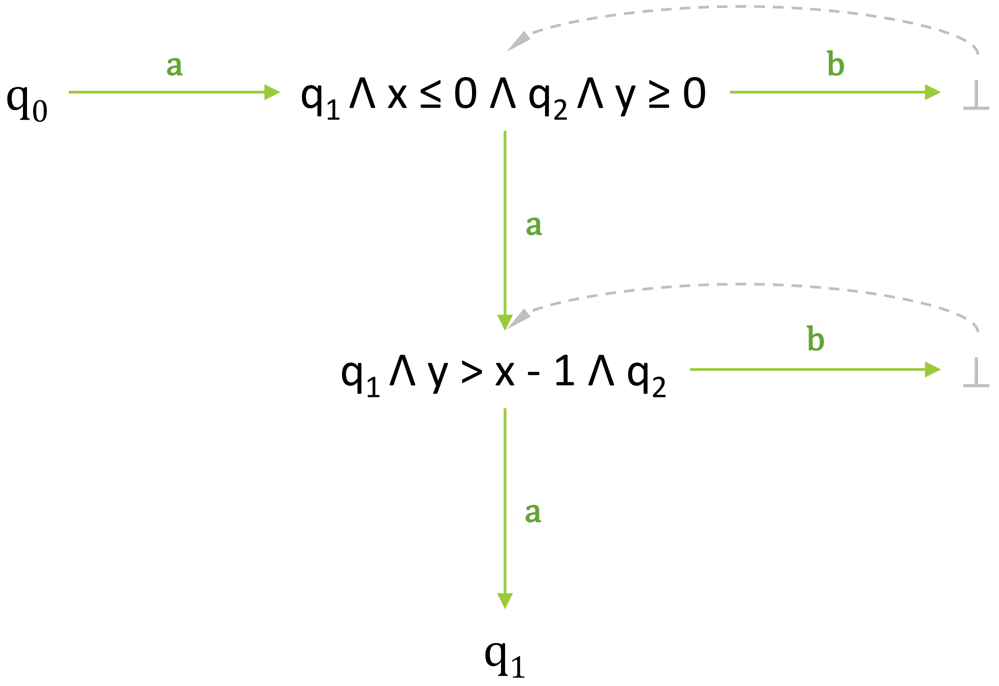
$\mathrm{ACC}(q_0, aab) = \perp$

Refine

Emptiness of Alternating Data Automata

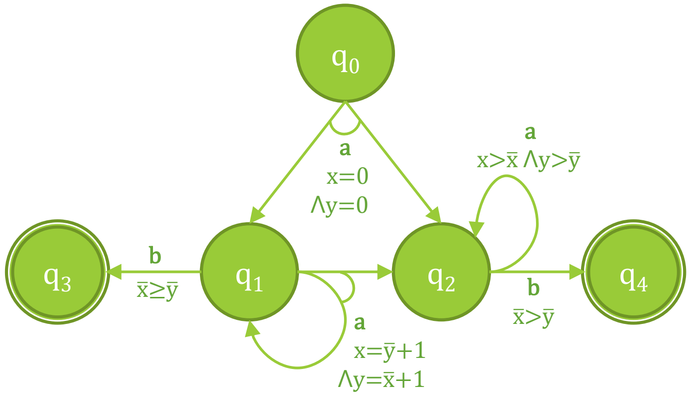
Impact

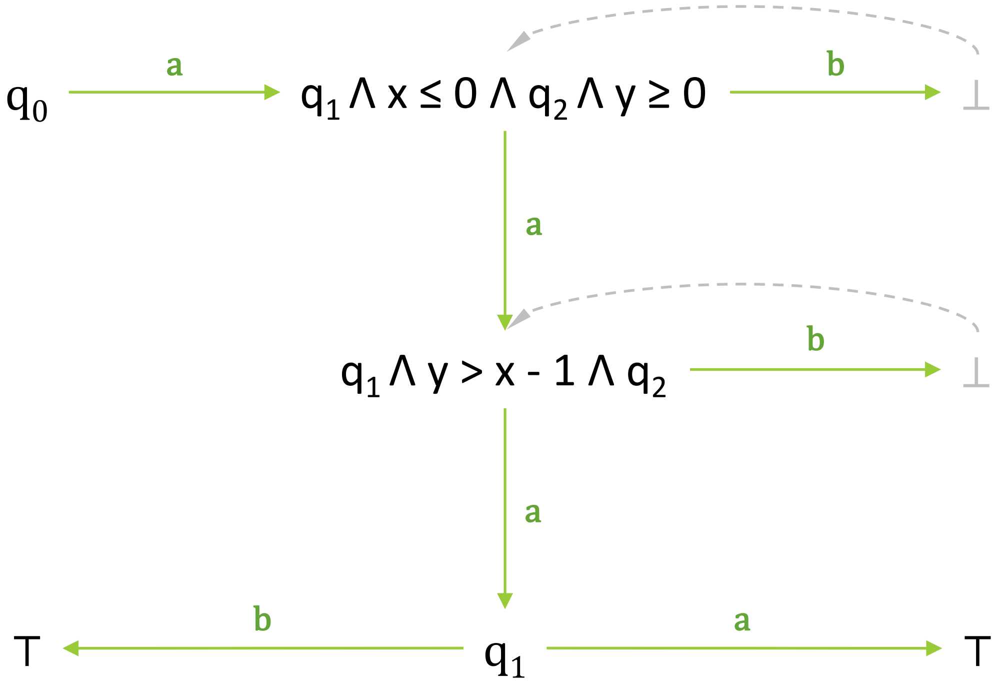




Emptiness of Alternating Data Automata

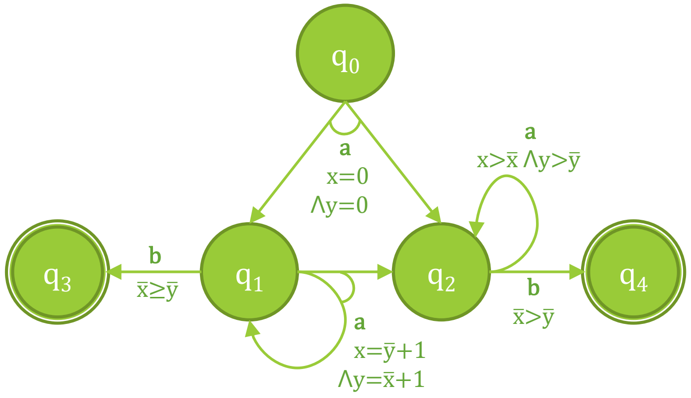
Impact

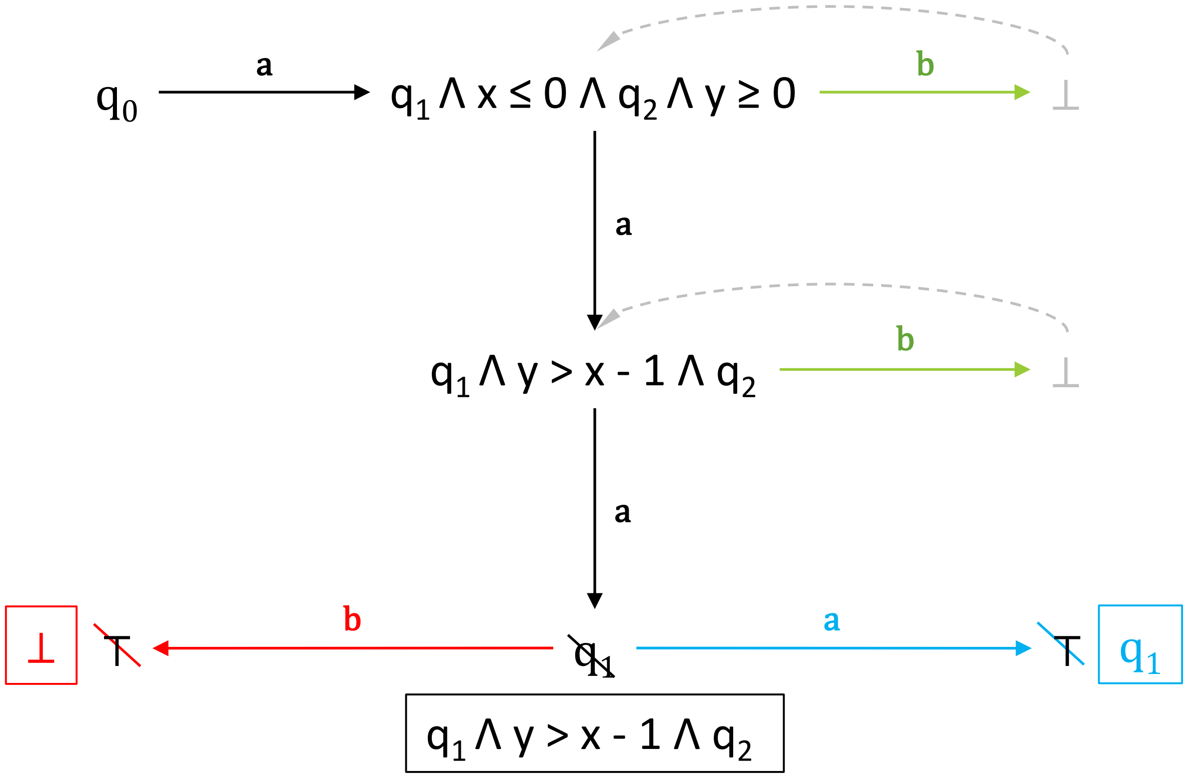




Emptiness of Alternating Data Automata

Impact





$ACC(q_0, aaaa) = \perp$

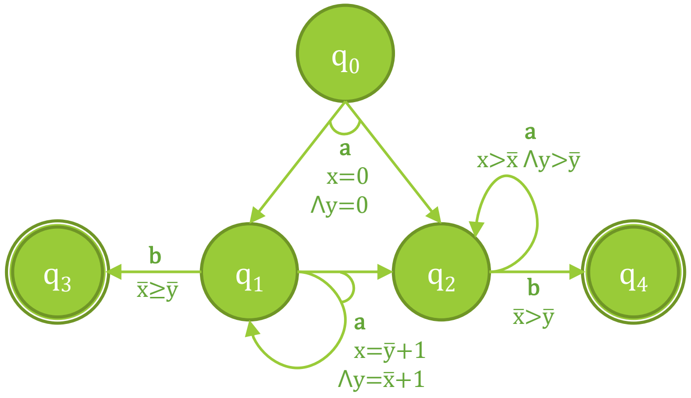
Refine

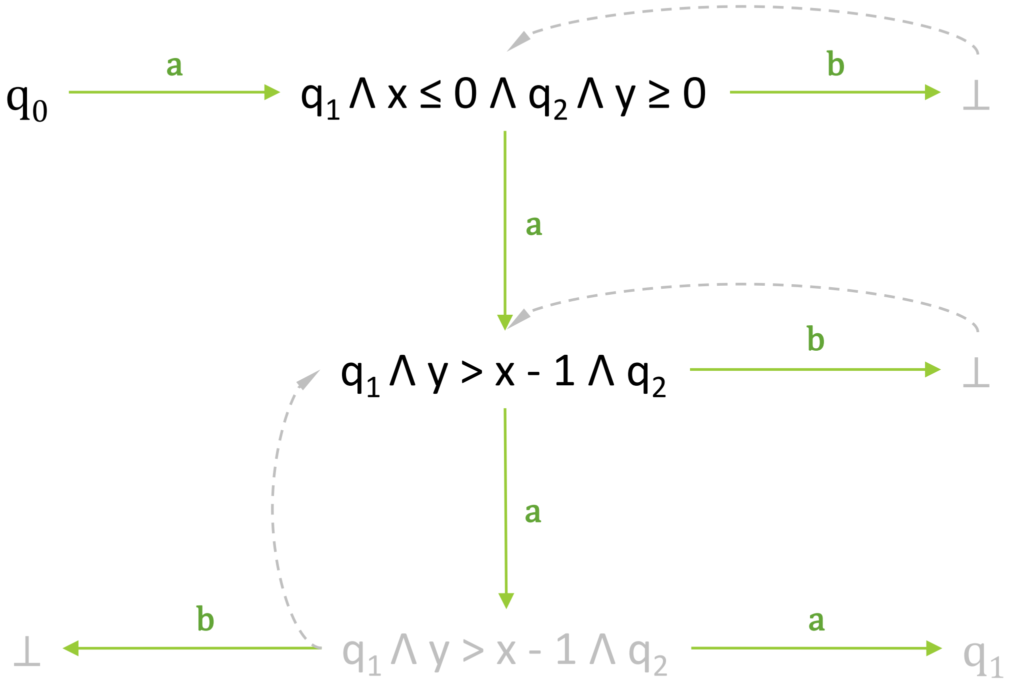
$ACC(q_0, aaab) = \perp$

Refine

Emptiness of Alternating Data Automata

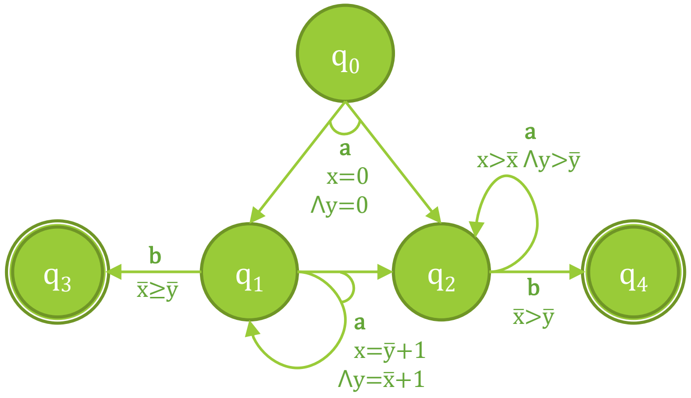
Impact

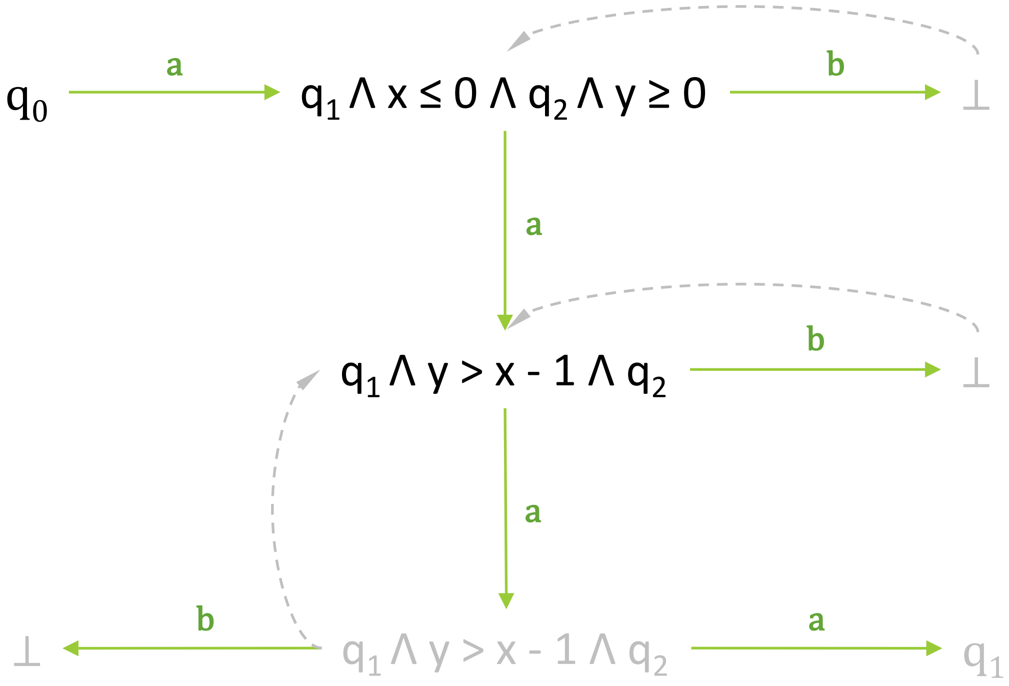




Emptiness of Alternating Data Automata

Impact





EMPTY

Emptiness of Alternating Data Automata

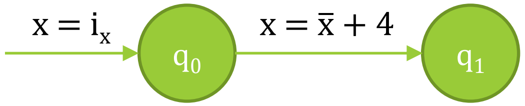
Experiments

Input Model	Size of Input (byte)	Empty?	Lazy Predicate Abstraction (sec)	Impact (sec)
simple1	321	no	0.576	0.061
simple2	520	yes	0.598	0.179
simple3	220	yes	0.702	0.144
array_shift	884	yes	1.88	0.199
array_simple	3456	yes	timeout	13.421
array_rotation	1848	yes	5.677	0.798
abp	6965	no	7.538	1.111
train	1875	yes	13.39	2.236
hw1	332	yes	1.101	0.22
hw2	690	yes	18.536	0.446

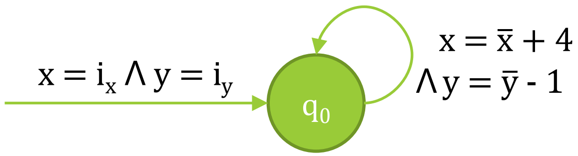
Future Work

Future Work

Ongoing Work



A



B

$L(A) \subseteq L(B)$ iff:

For each word $w = (a_1, \langle x_1 \rangle)(a_2, \langle x_2 \rangle)\ldots(a_k, \langle x_k \rangle)$ in $L(A)$, there exists a series of data $Y = y_1, y_2, \ldots, y_k$ such that $(a_1, \langle x_1, y_1 \rangle)(a_2, \langle x_2, y_2 \rangle)\ldots(a_k, \langle x_k, y_k \rangle)$ is in $L(B)$.

Related Works and References

1. Lazy Abstraction with Interpolants
   Kenneth L. McMillan, Cadence Berkeley Labs
   Computer Aided Verification, Volume 4144 of the series Lecture Notes in Computer Science pp 123-136

2. When Simulation Meets Antichains
   Parosh Aziz Abdulla, Yu-Fang Chen, Lukáš Holík, Richard Mayr, Tomáš Vojnar
   TACAS 2010: Tools and Algorithms for the Construction and Analysis of Systems pp 158-174

3. Antichain Algorithms for Finite Automata
   Laurent Doyen, Jean-François Raskin
   TACAS 2010: Tools and Algorithms for the Construction and Analysis of Systems pp 2-22

4. Alternation
   ASHOK K. CHANDRA, DEXTER C. KOZEN, AND LARRY J. STOCKMEYER
   Foundations of Computer Science, 1976., 17th Annual Symposium

5. Alternating Timed Automata
   Slawomir Lasota, Igor Walukiewicz
   ACM Transactions on Computational Logic (TOCL), Volume 9 Issue 2, March 2008, Article No. 10

6. Tree Interpolation in Vampire
   Regis Blanc, Ashutosh Gupta, Laura Kovacs, Bernhard Kragl
   Logic for Programming, Artificial Intelligence, and Reasoning Volume 8312 of the series Lecture Notes in Computer Science pp 173-181