Modeling Systems and Specifications over Infinite Data

Hadar Frenkel[1]

Joint work with Orna Grumberg[1] and Sarai Sheinvald[2]

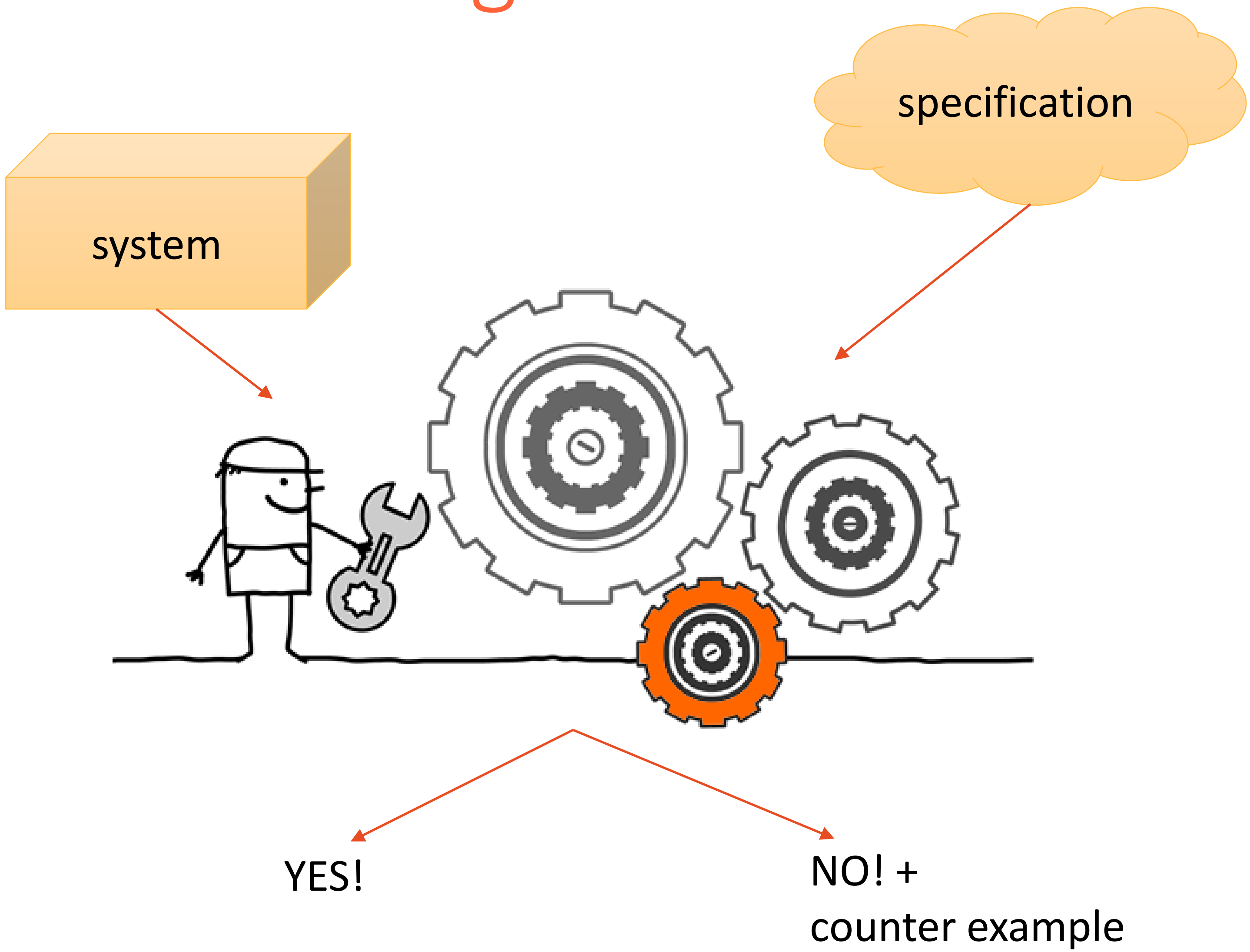
(1) Technion - Israel Institute of Technology

(2) Braude college of engineering, Karmiel, Israel

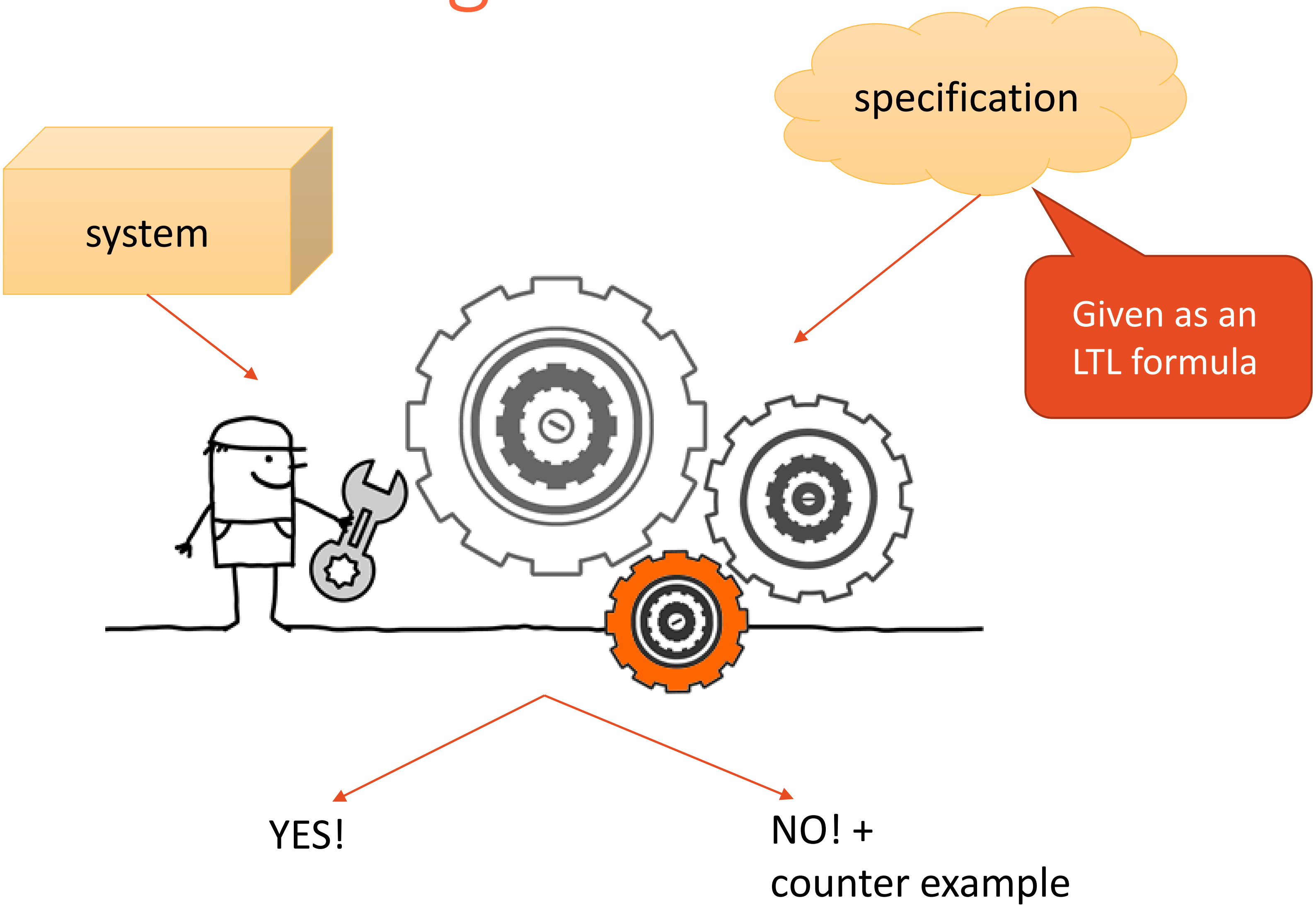
Goal

- Develop a Model checking process for systems over infinite data domains
- using the automata-theoretic approach

Model checking



Model checking



LTL Specifications

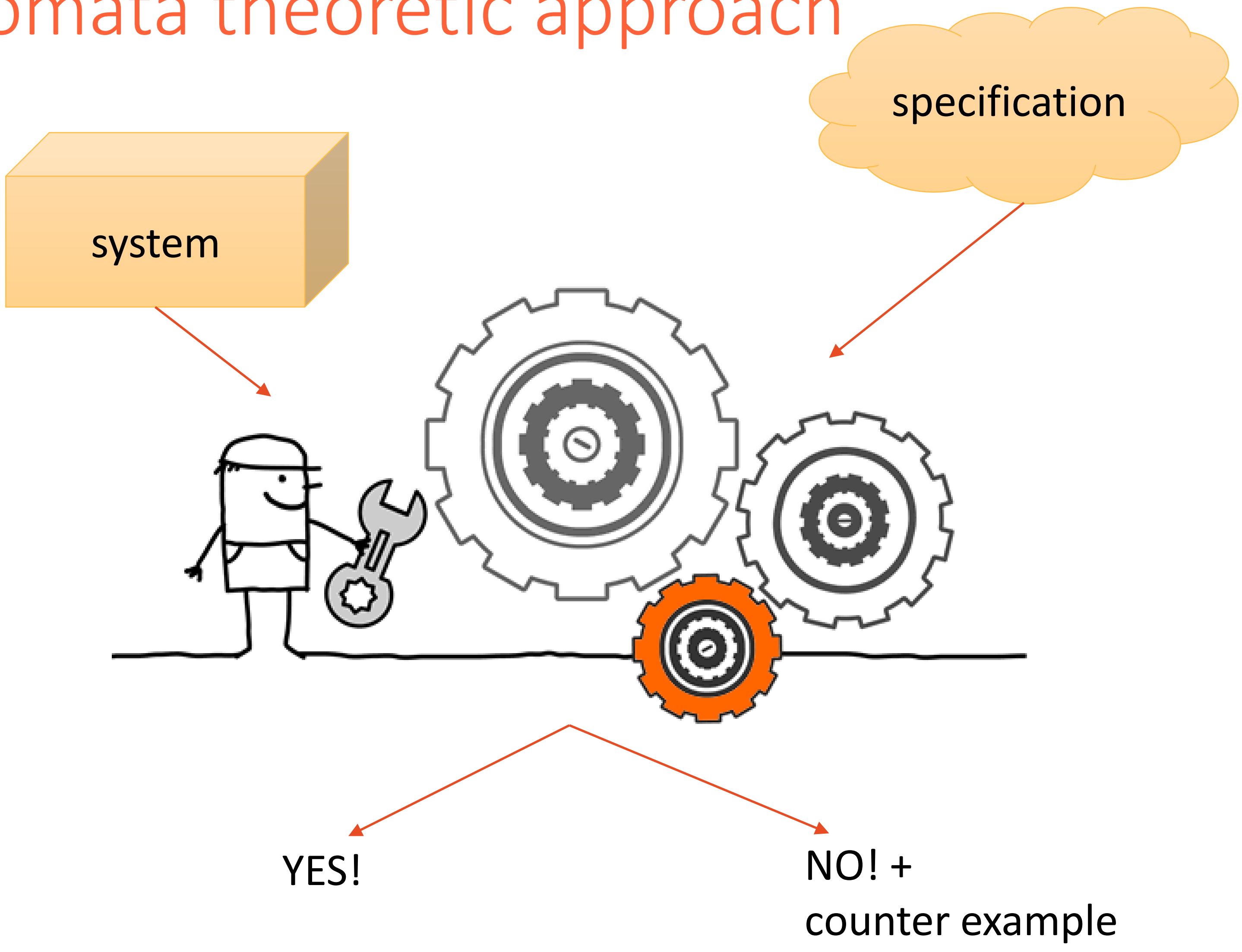
- **L**inear **T**emporal **L**ogic formulas.
- Describe occurrences of events during time

- E.g.:

$$G(send \rightarrow F\ receive)$$

- Send, send, receive, send, receive,...

Model checking - automata theoretic approach

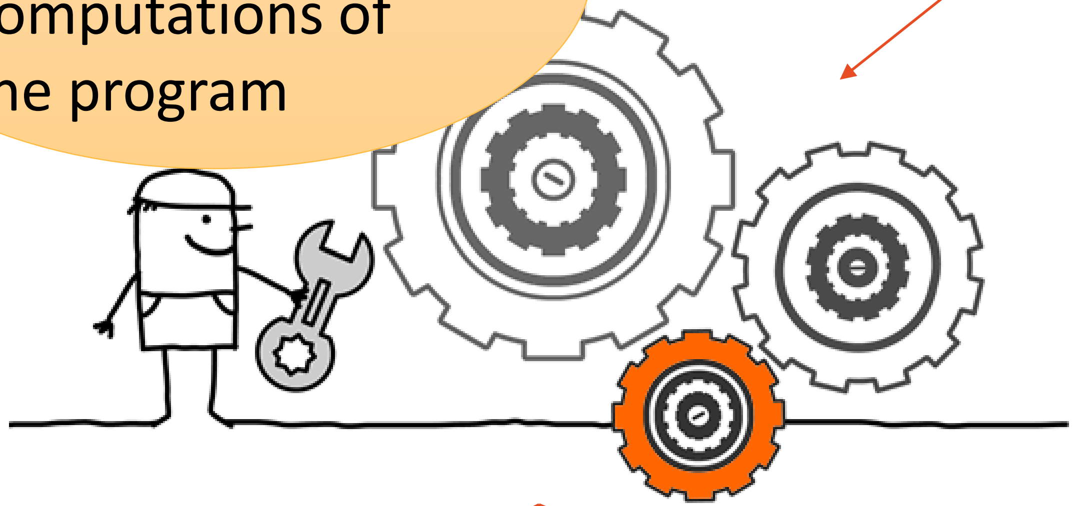


Model checking - automata theoretic approach

specification

A_P

Recognizes all legal computations of the program



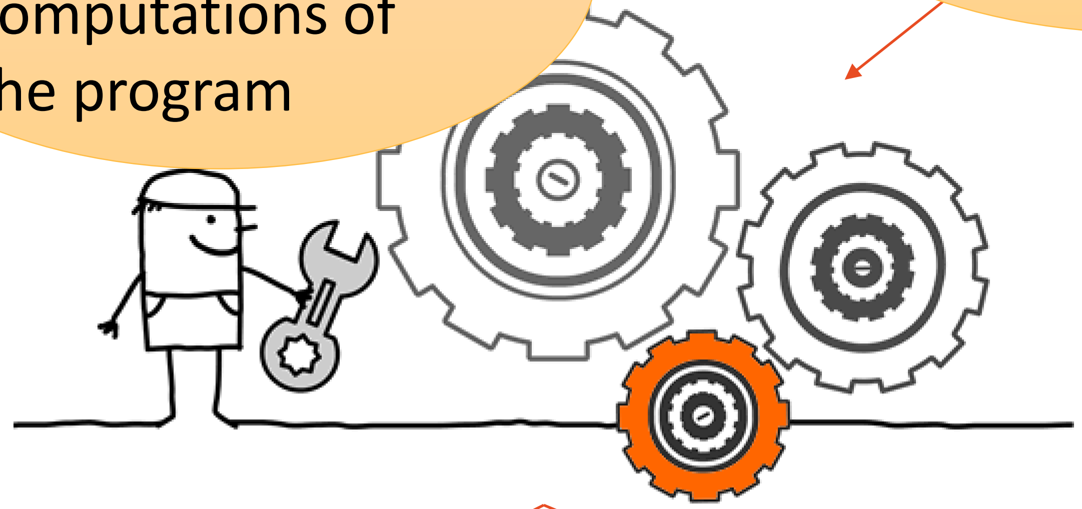
YES!

NO! +
counter example

Model checking - automata theoretic approach

A_P
Recognizes all legal computations of the program

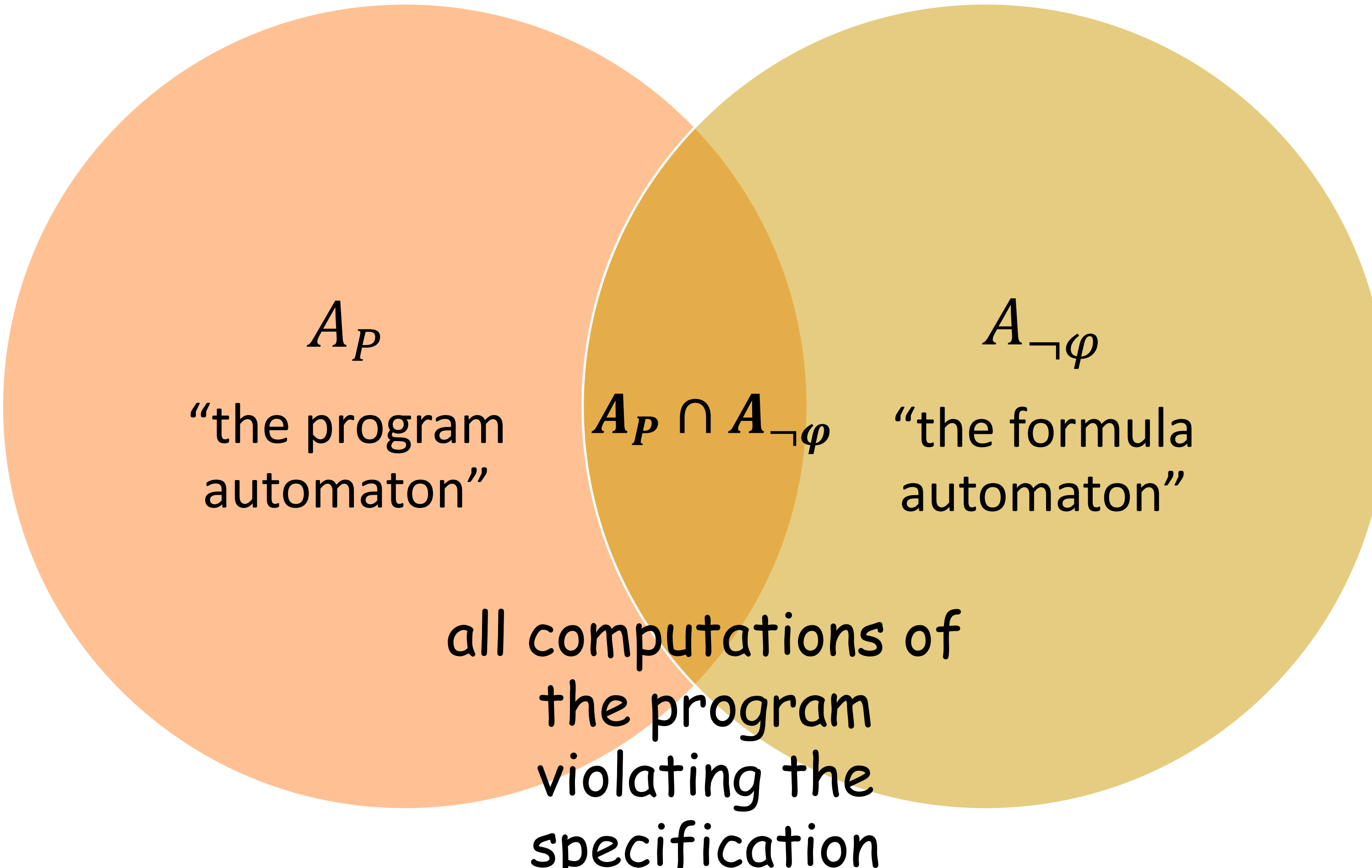
$A_{\neg\varphi}$
Recognizes the set of computations violating φ



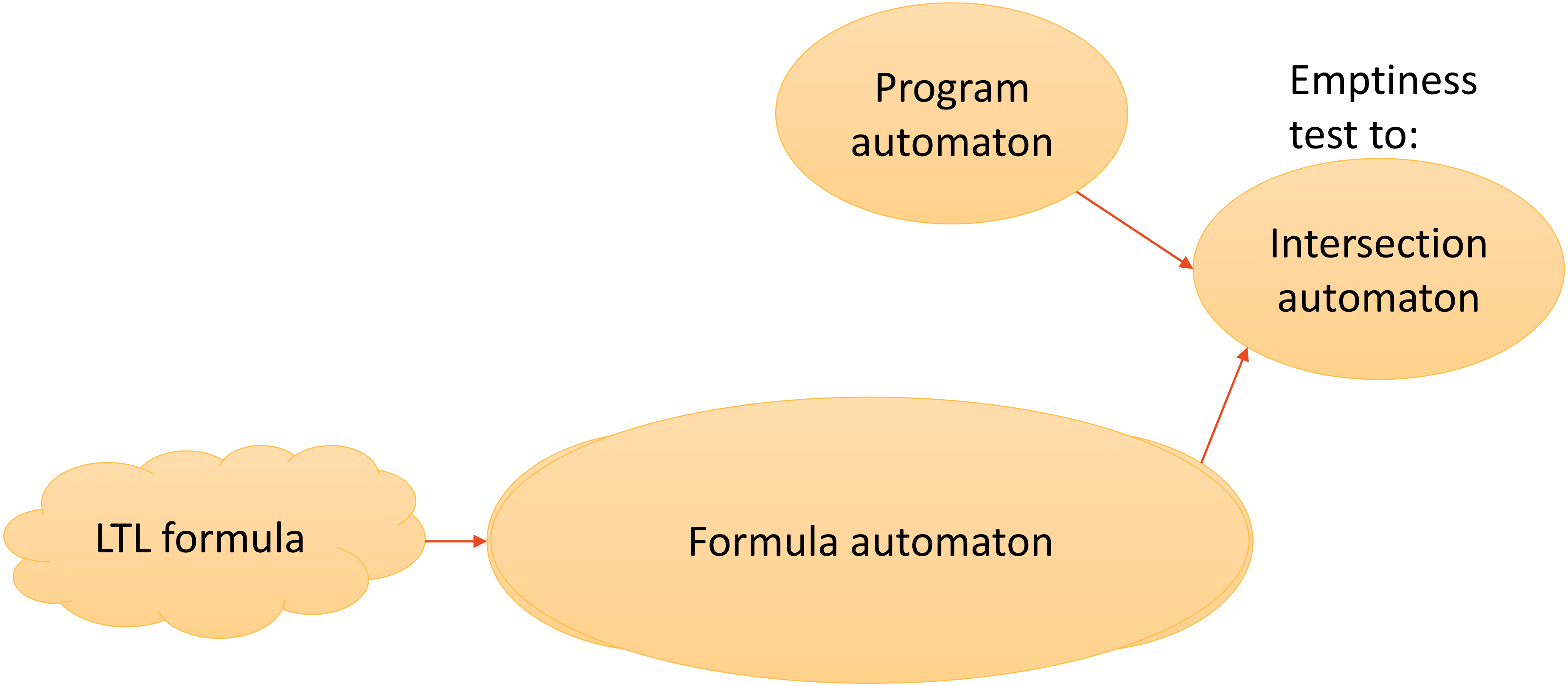
YES!

NO! +
counter example

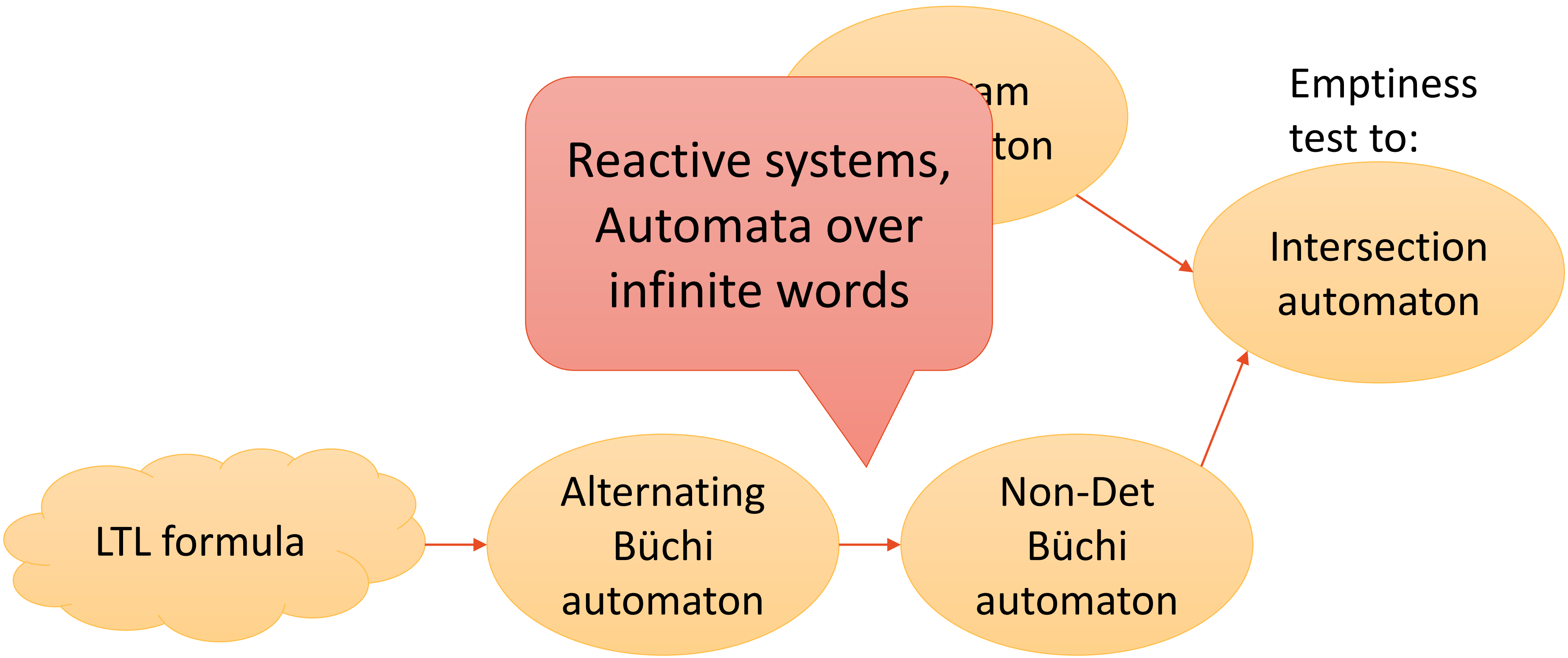
Model checking - automata theoretic approach



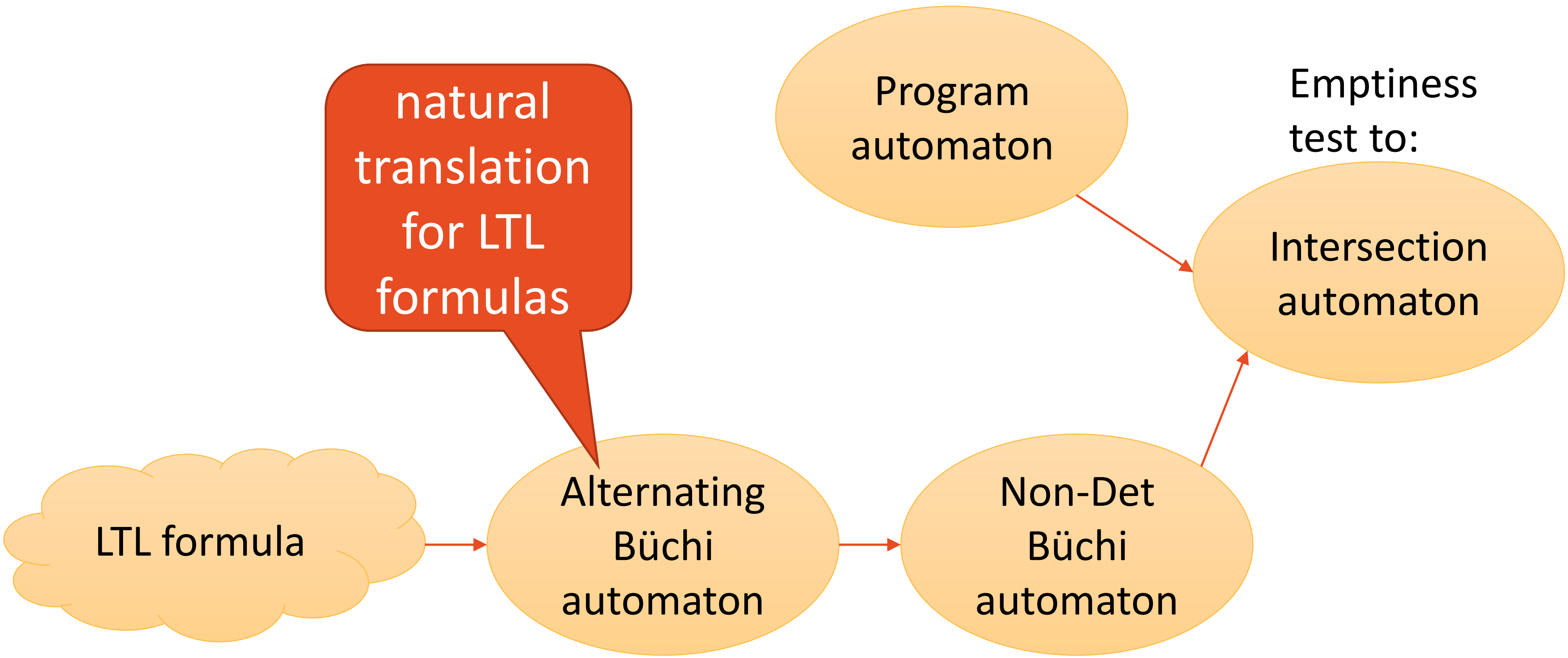
Model checking - automata theoretic approach



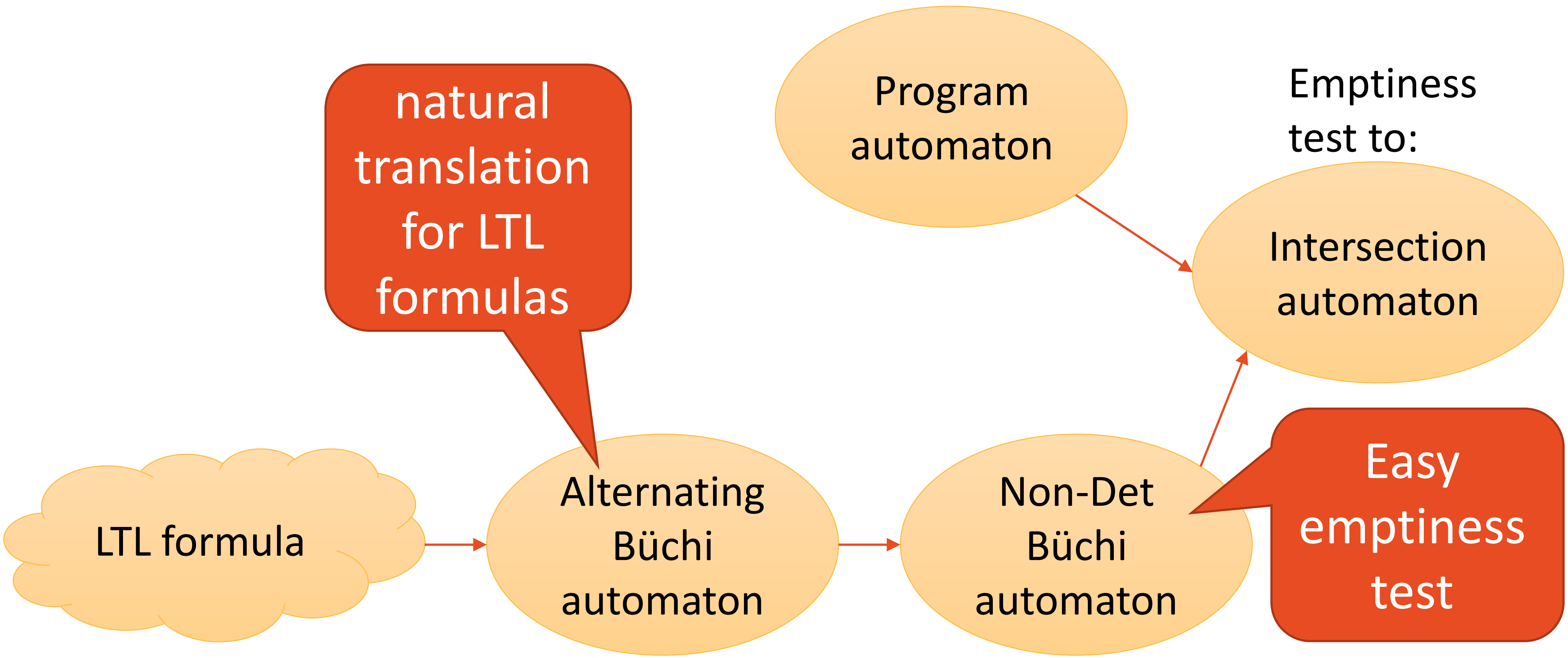
Model checking - automata theoretic approach



Model checking - automata theoretic approach



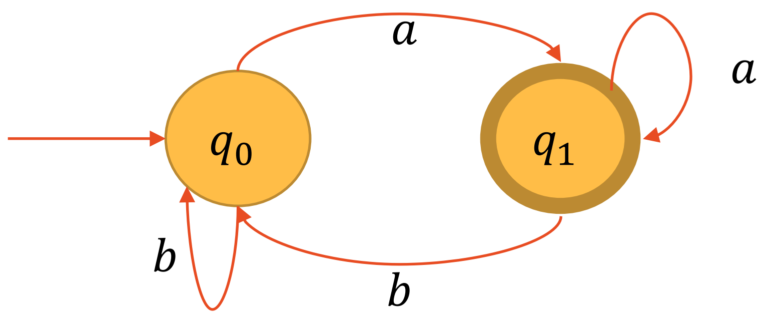
Model checking - automata theoretic approach



Non-Deterministic Büchi automaton Over Infinite Words

- $L = \{w \in \{a, b\}^{\omega} \mid w \text{ has infinite number of } a\text{'s}\}$

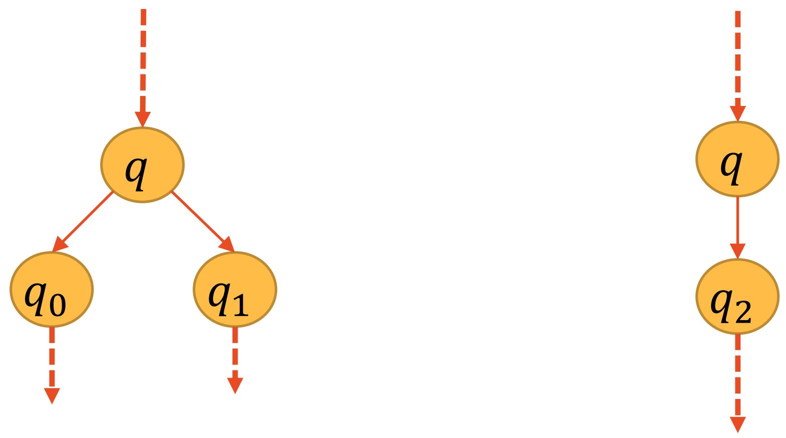
- Büchi automaton:



- Visit an accepting state infinitely often

Alternating Büchi Automaton Over Infinite Words

- Transition function is a positive boolean expression over states
- $\delta(q, a) = (q_0 \wedge q_1) \vee q_2$



Run #1

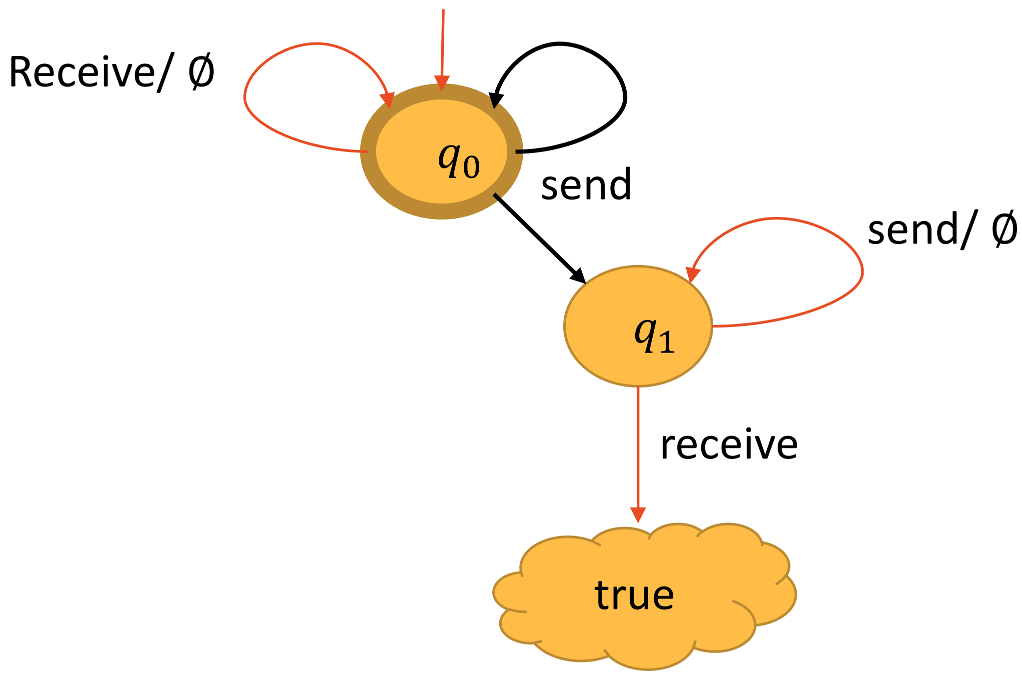
Run #2

- *Each infinite path* has to visit an accepting state infinitely often

The Formula Automaton [V96]

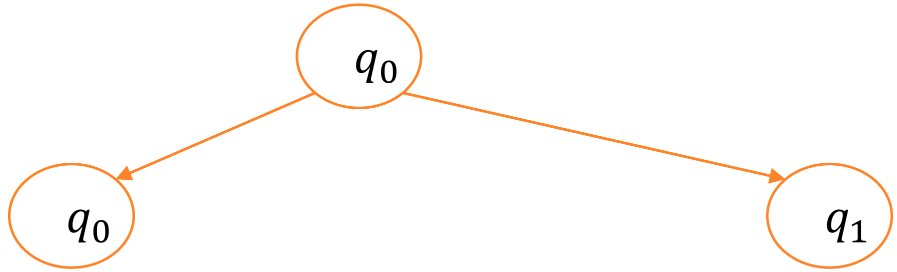
- A linear translation of an LTL formula to an alternating Büchi automaton

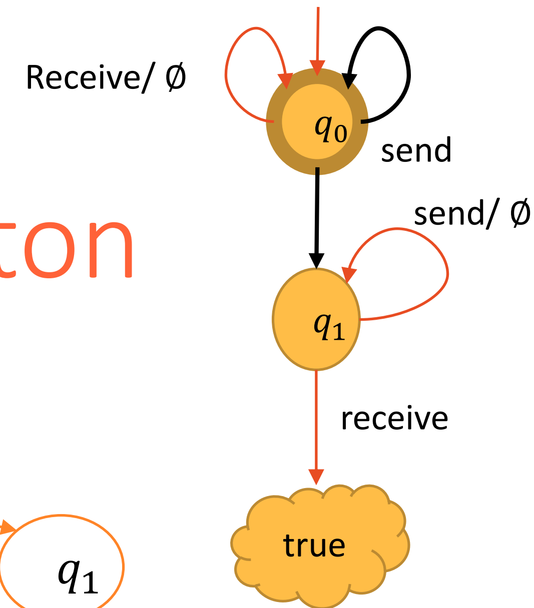
- *G (send→ XF receive)*



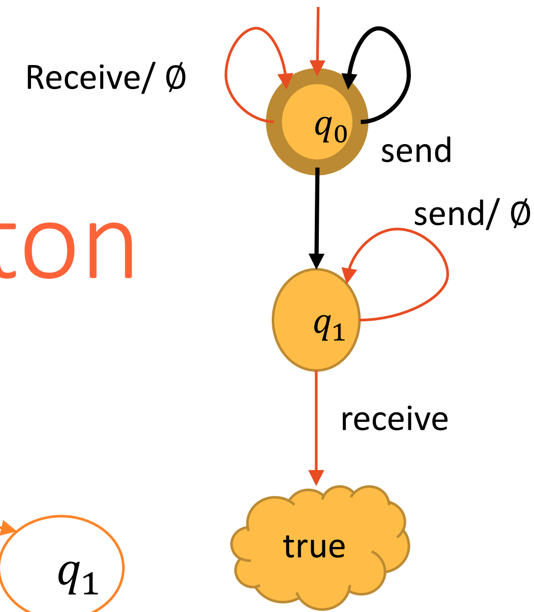
Example - a Run of an Alternating Büchi automaton

Send

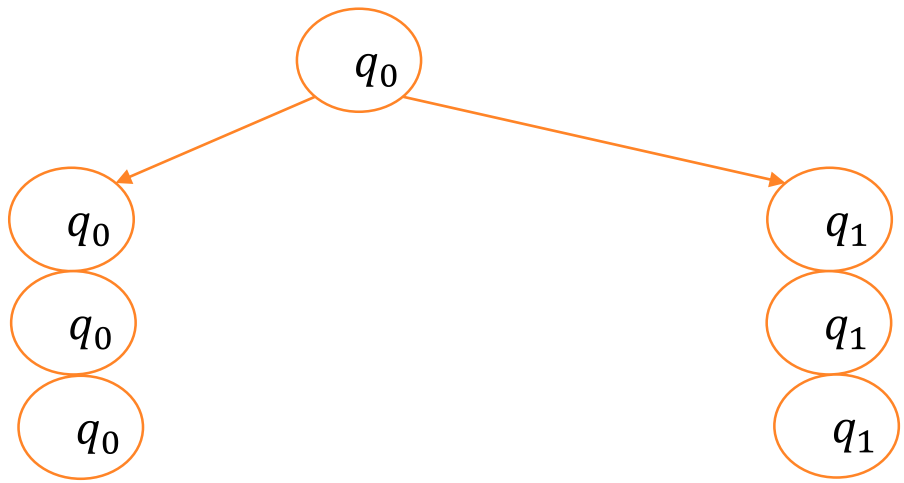




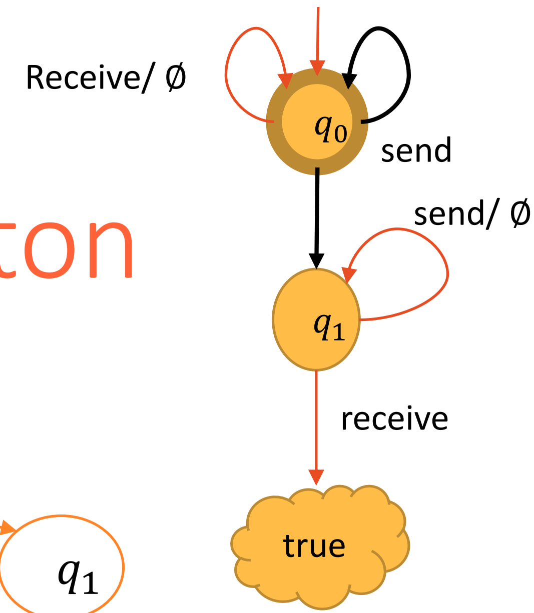
Example - a Run of an Alternating Büchi automaton



Send



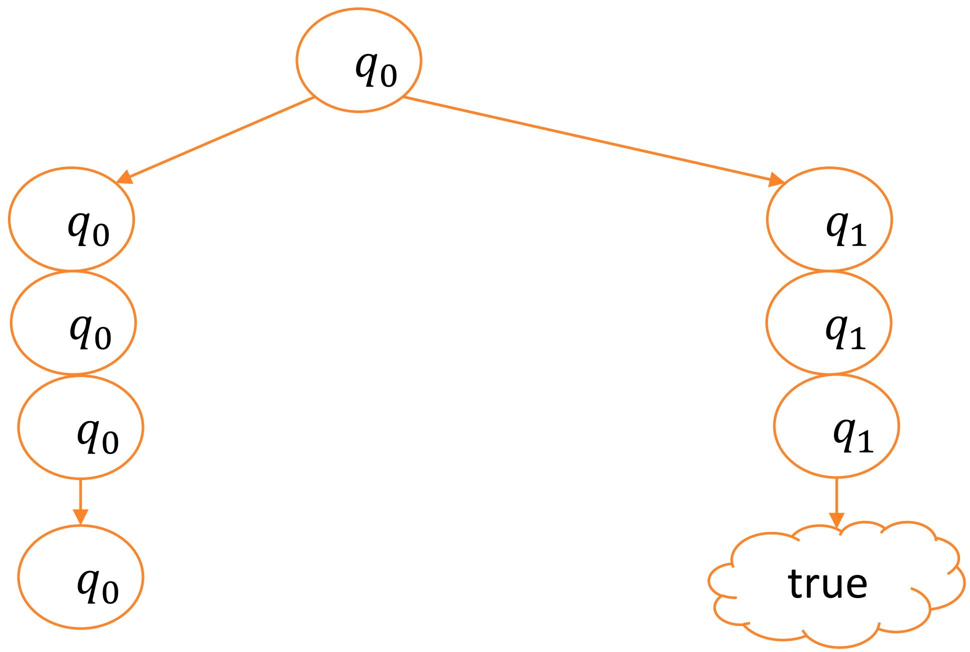
Example - a Run of an Alternating Büchi automaton



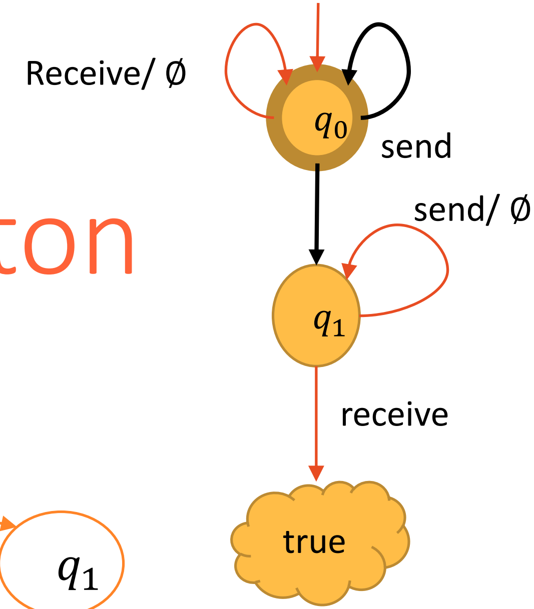
Send

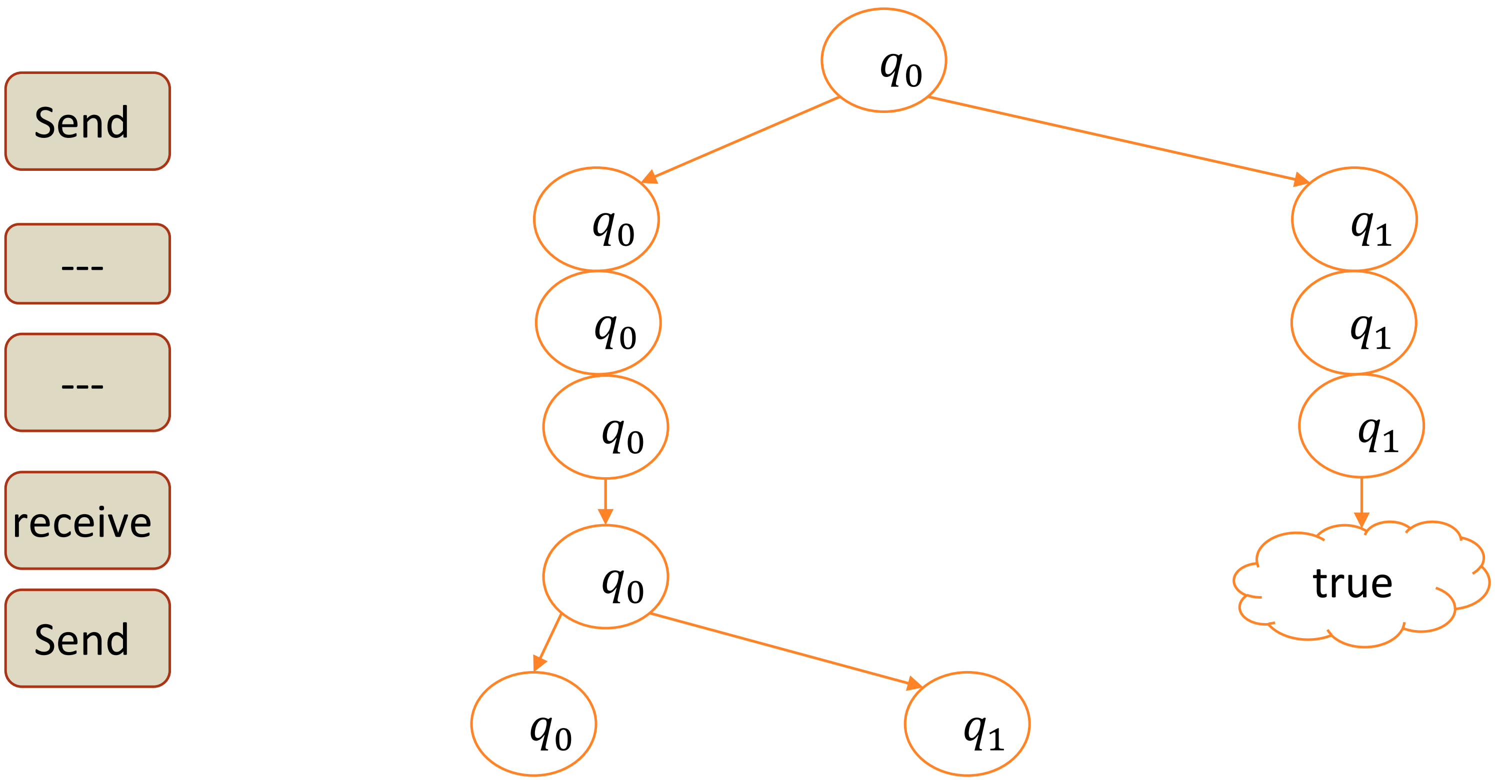
---

receive

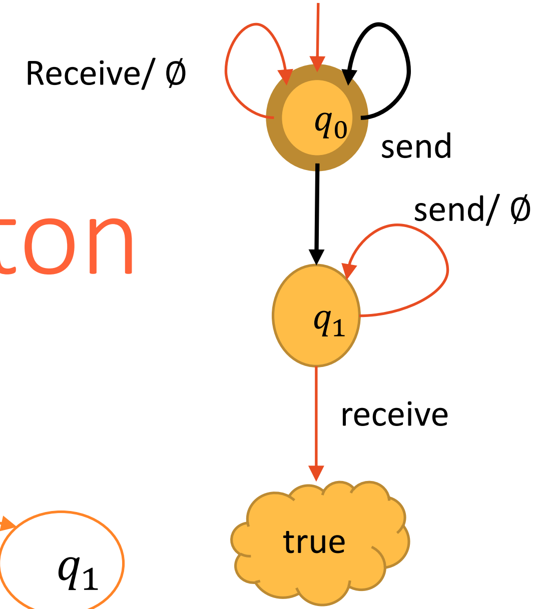


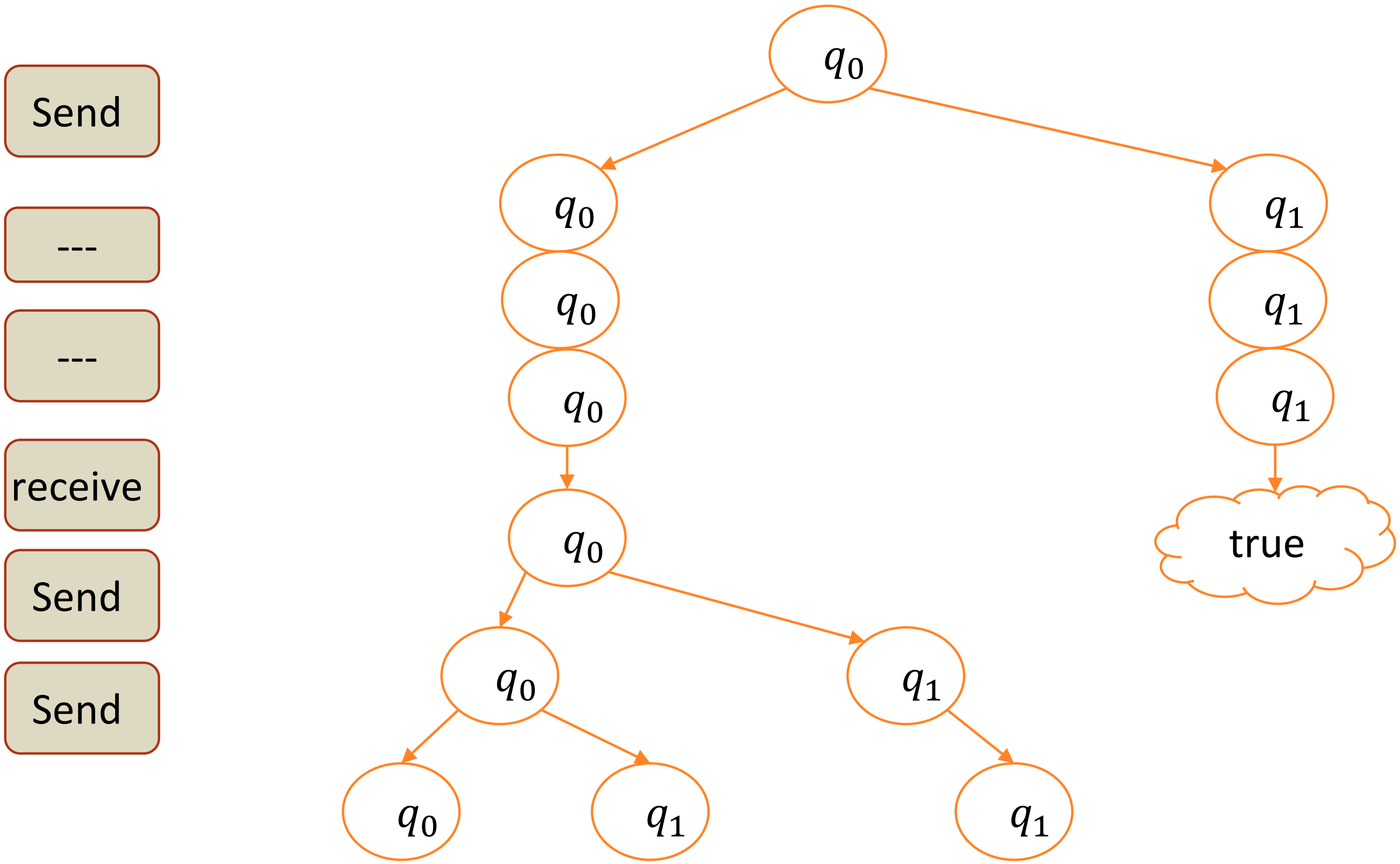
Example - a Run of an Alternating Büchi automaton



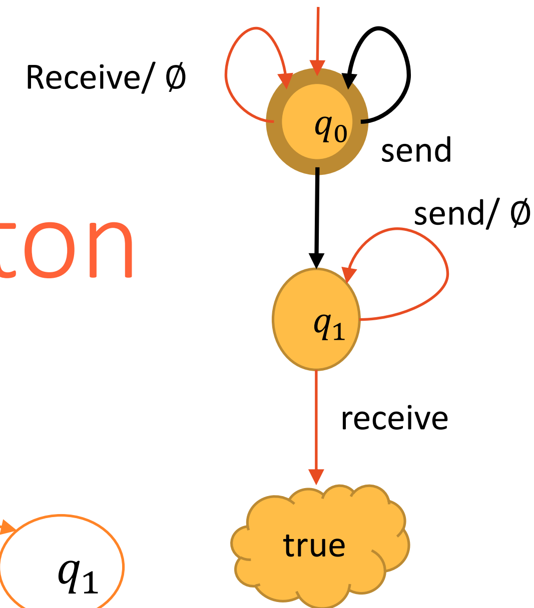


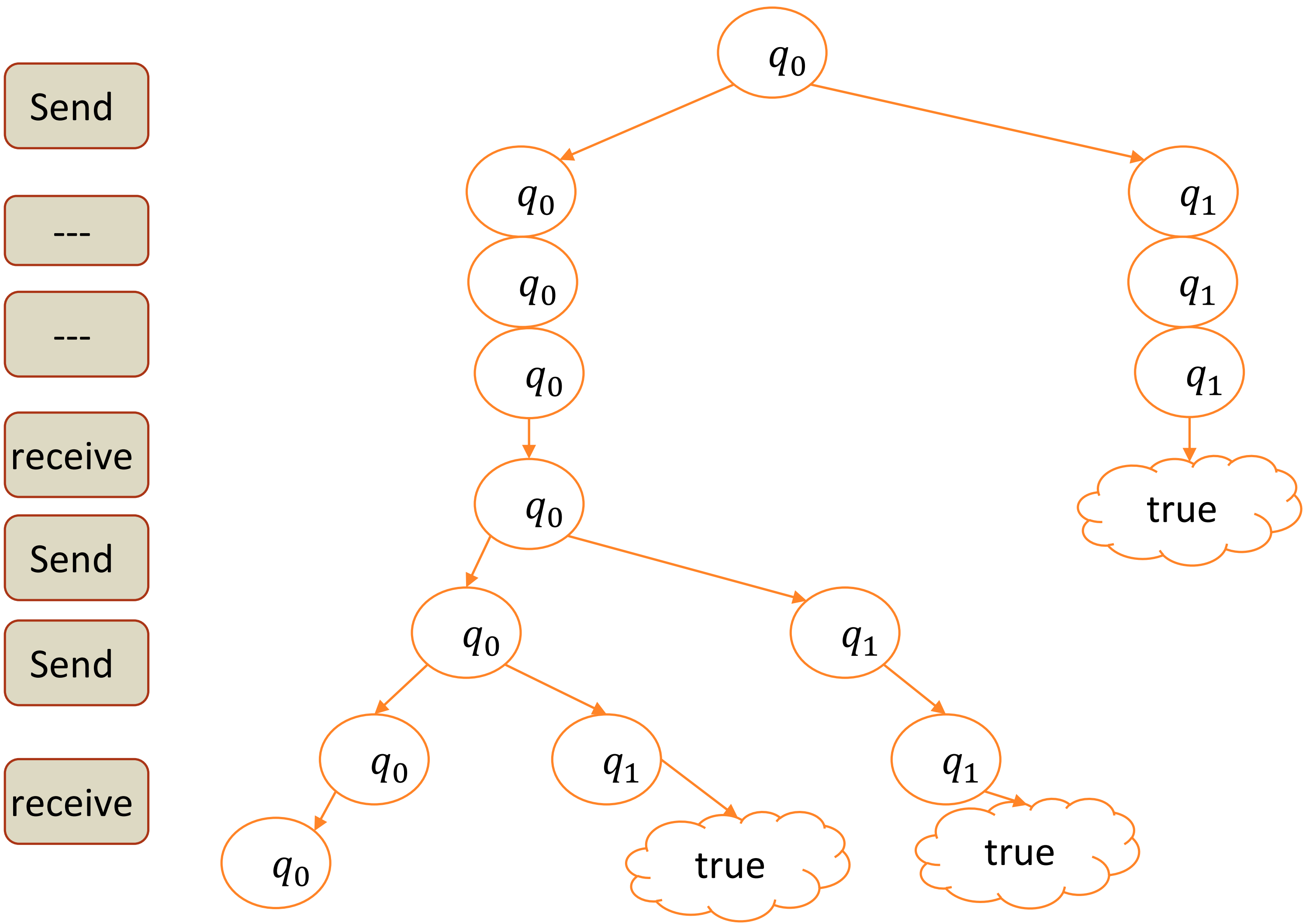
Example - a Run of an Alternating Büchi automaton





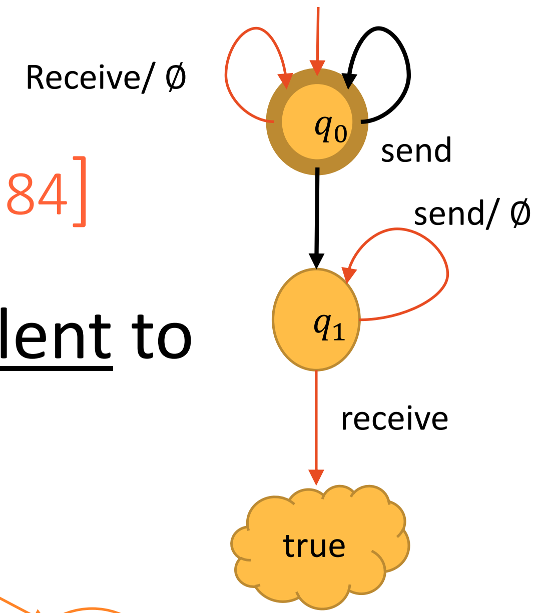
Example - a Run of an Alternating Büchi automaton

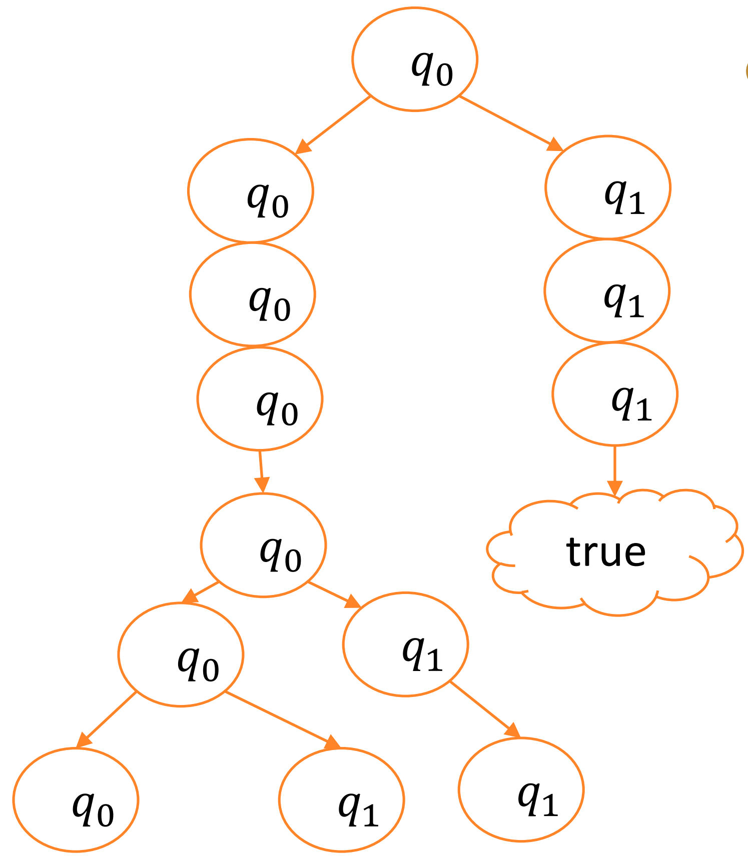




Alternating to Non-Det [MH84]

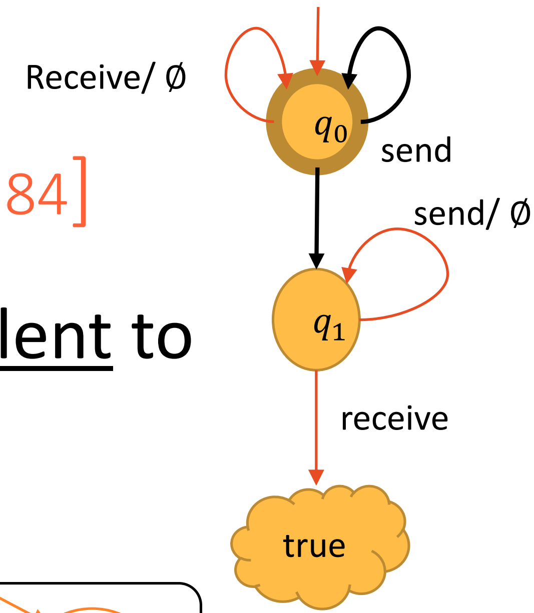
- Alternating Büchi automata are <u>equivalent</u> to non-deterministic Büchi automata

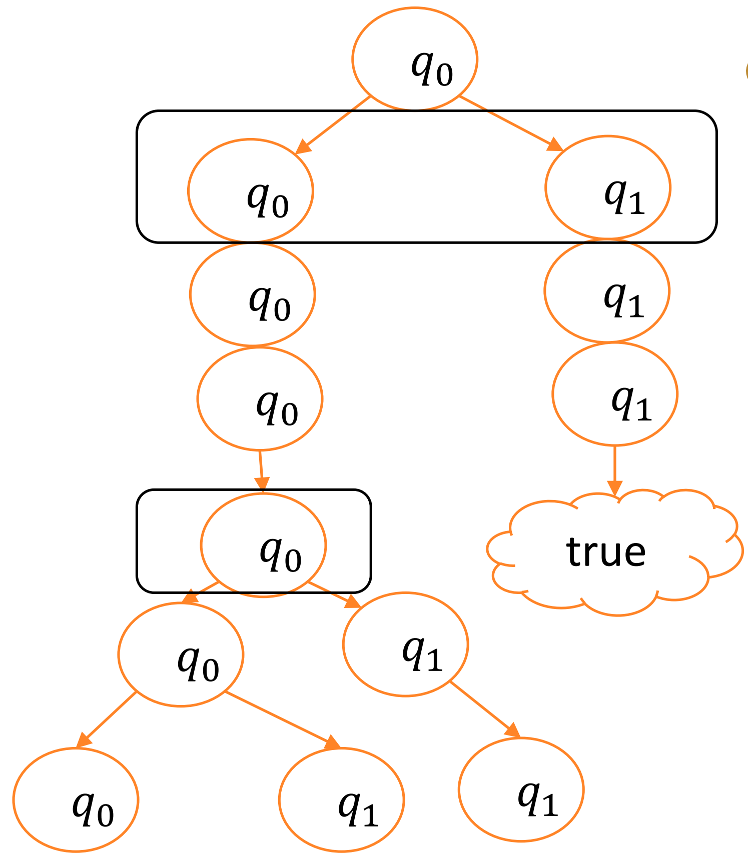




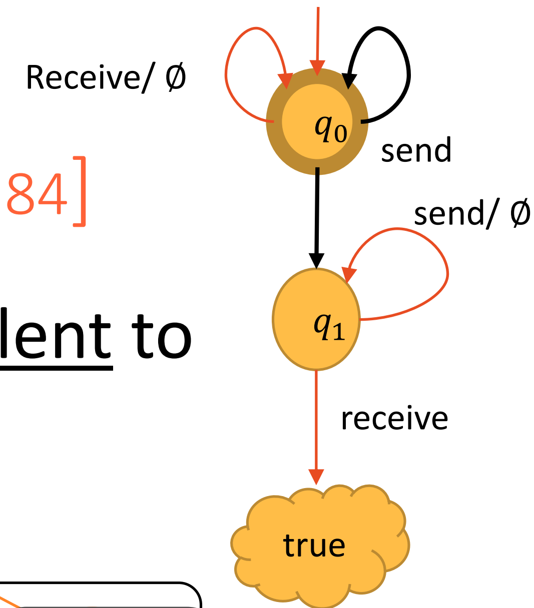
Alternating to Non-Det [MH84]

- Alternating Büchi automata are <u>equivalent</u> to non-deterministic Büchi automata

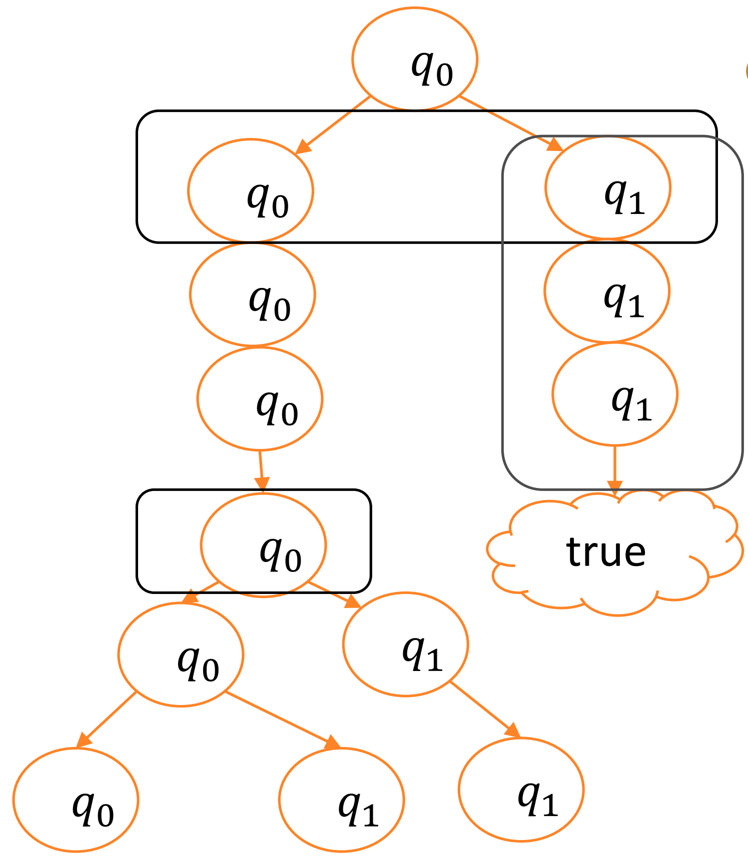




Alternating to Non-Det [MH84]

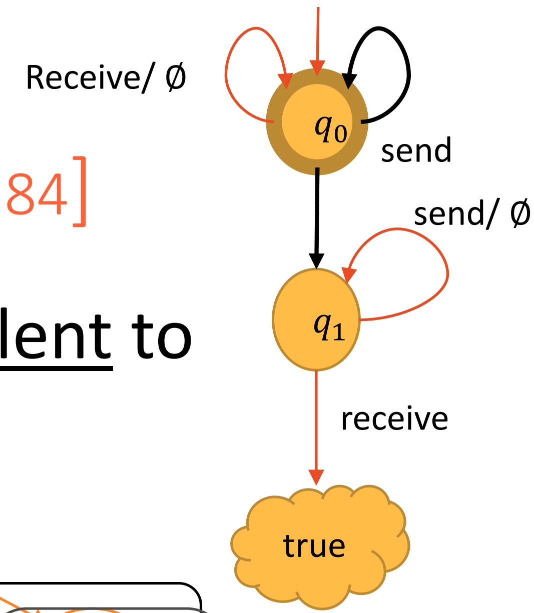


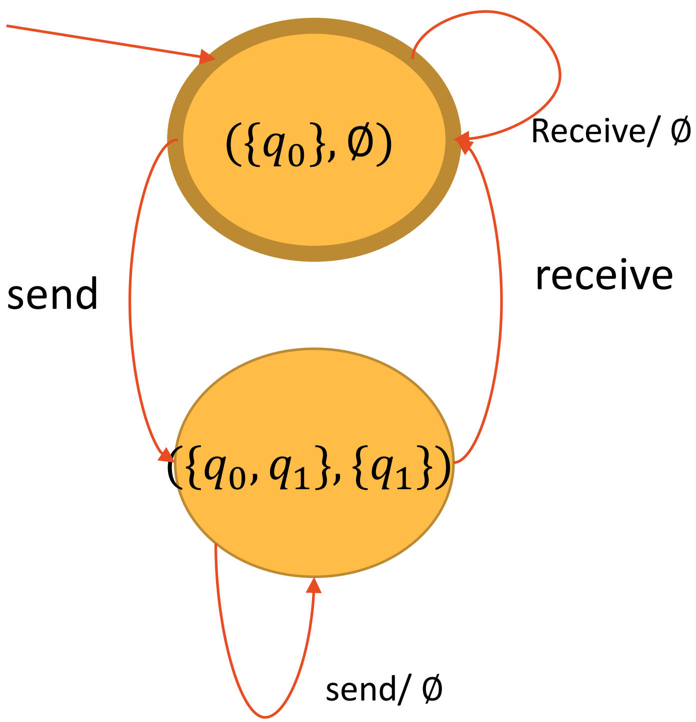
- Alternating Büchi automata are <u>equivalent</u> to non-deterministic Büchi automata

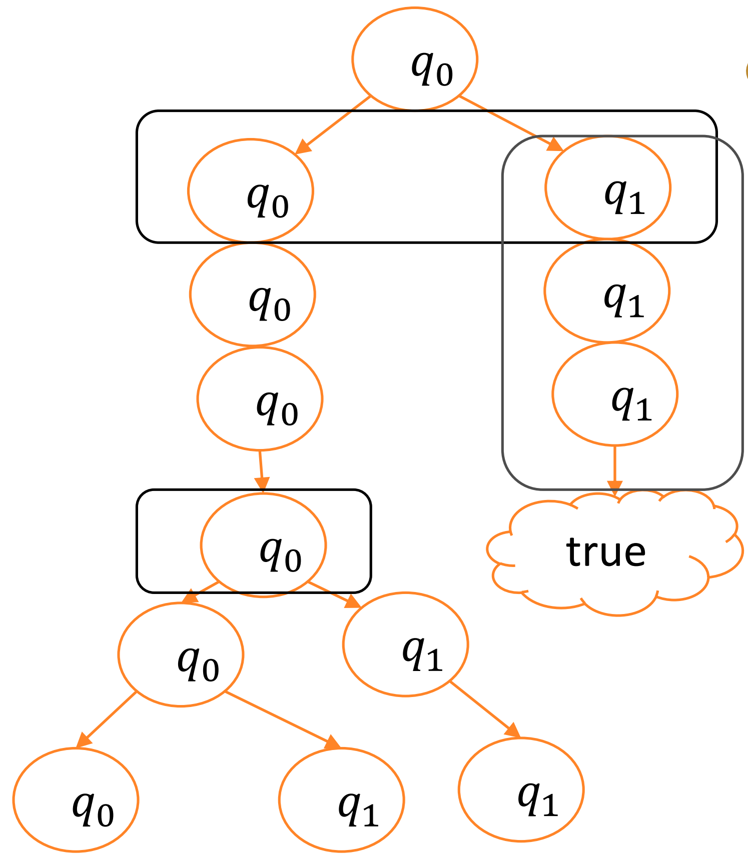


Alternating to Non-Det [MH84]

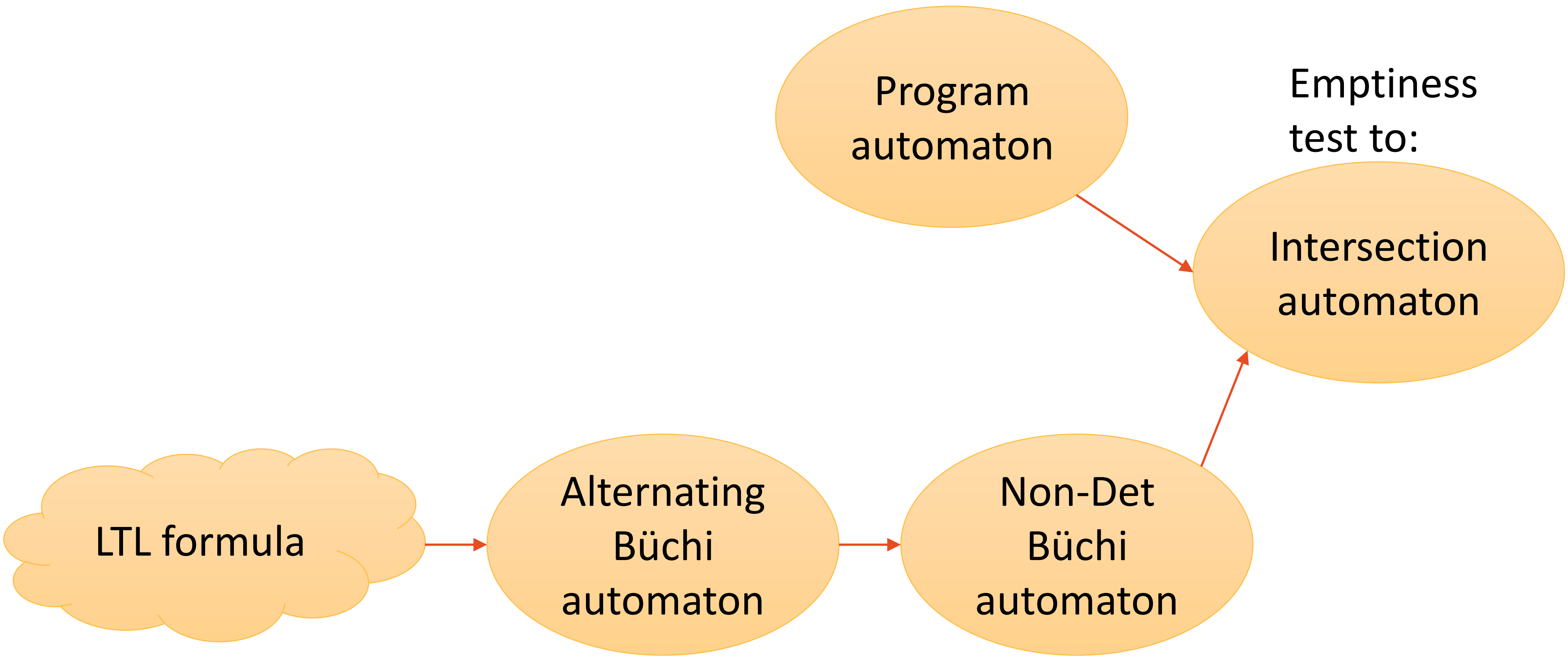
- Alternating Büchi automata are equivalent to non-deterministic Büchi automata







Summary: Model checking - automata theoretic approach



Our Setting

- **An infinite** (or as large as we want) data domain

- Motivation: dynamic creation (and deletion) of processes, messages arriving…
- E.g.: server and clients, the set of clients is unknown in advance.
- “every client is eventually active”
    - LTL cannot express this property

Variable LTL [GKS12]

- Example:

$\forall x: F\ active.x$

- AP - finite set of propositions
- V - finite set of variables
- Parameterized propositions instead of atomic propositions
- Quantifiers

$\exists^*$-VLTL

- VLTL with only existential quantifiers
- $G\ \exists x: send.x$
- A possible satisfying computation:

$send.1$

$send.1$

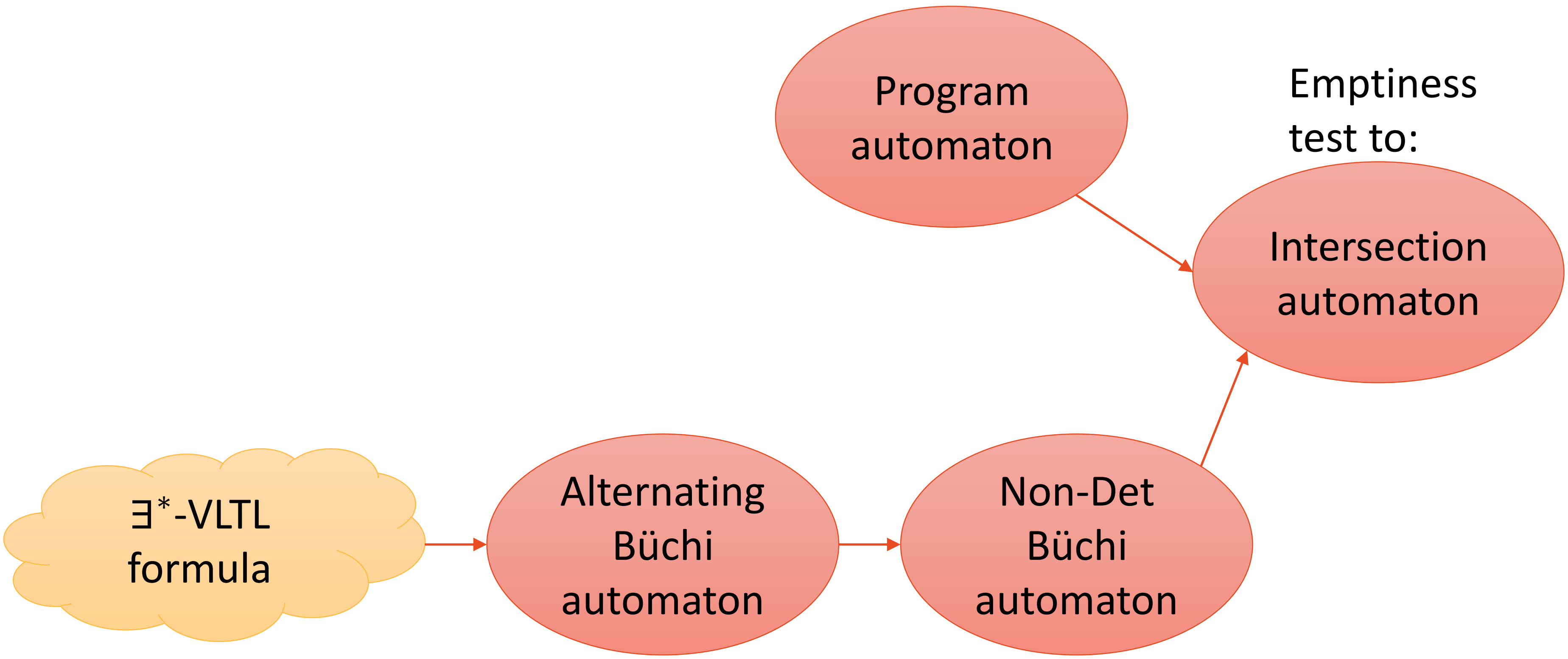
$send.4$

$send.7$

$send.3$

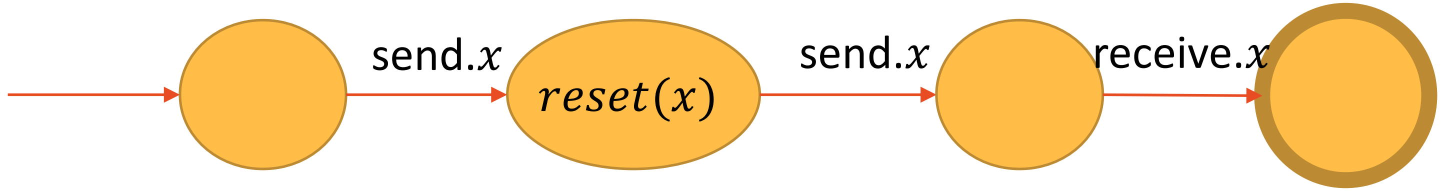
$send.9$

Model checking Process
Infinite Data Domains



Automata With Variables [GKS12]

- Variables (or parameterized propositions) as the alphabet
- Ability to reset a variable: “forget” its value and assign a new value
- As long as there is no reset - the value cannot be changed



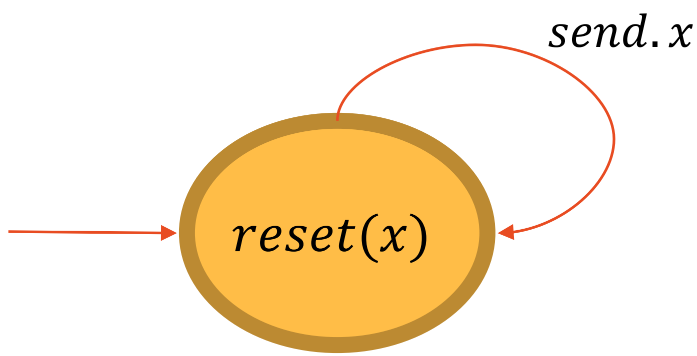
Satisfying computations:

send.1, send.2, receive.2

send.3, send.8, receive.8

Non-Det Büchi Automaton With Variables

- $G\ \exists x\colon send.x$



$send.1$

$send.1$

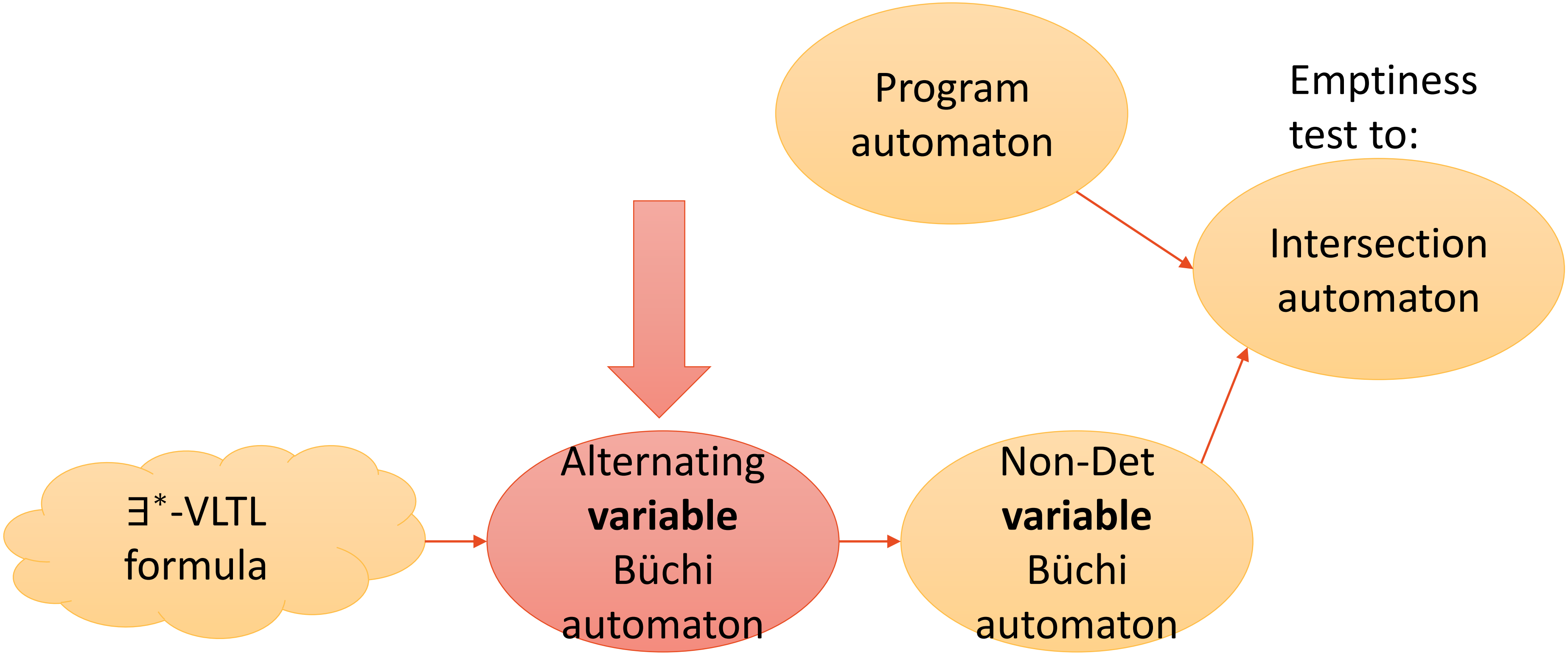
$send.4$

$send.7$

$send.3$

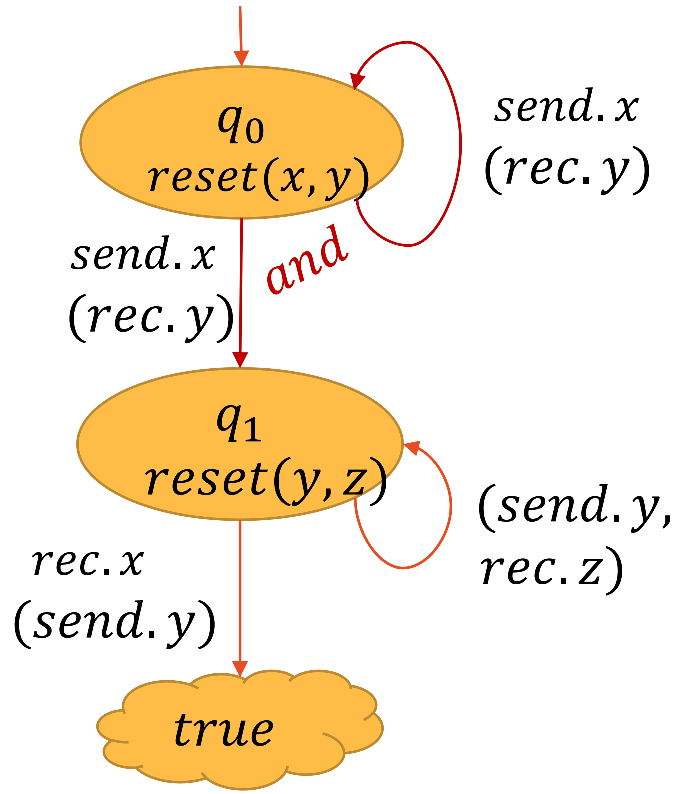
$send.9$

Model checking Process
Infinite Data Domains



Alternating Büchi Variable Automata [FGS17]

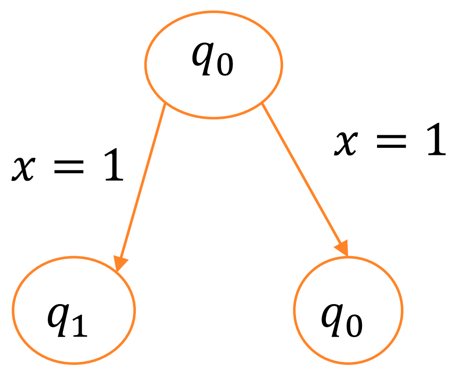
- Similar semantics as non-deterministic
- $G\ \exists x(send.x \wedge XF\ rec.x)$

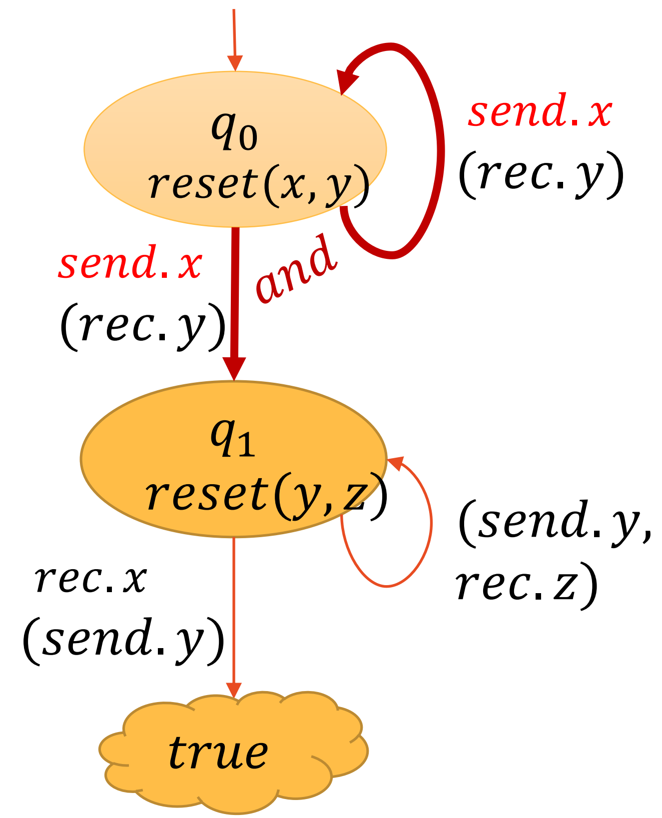


Alternating Variable Büchi Automata

- $G \exists x (send.x \wedge XF\, rec.x)$

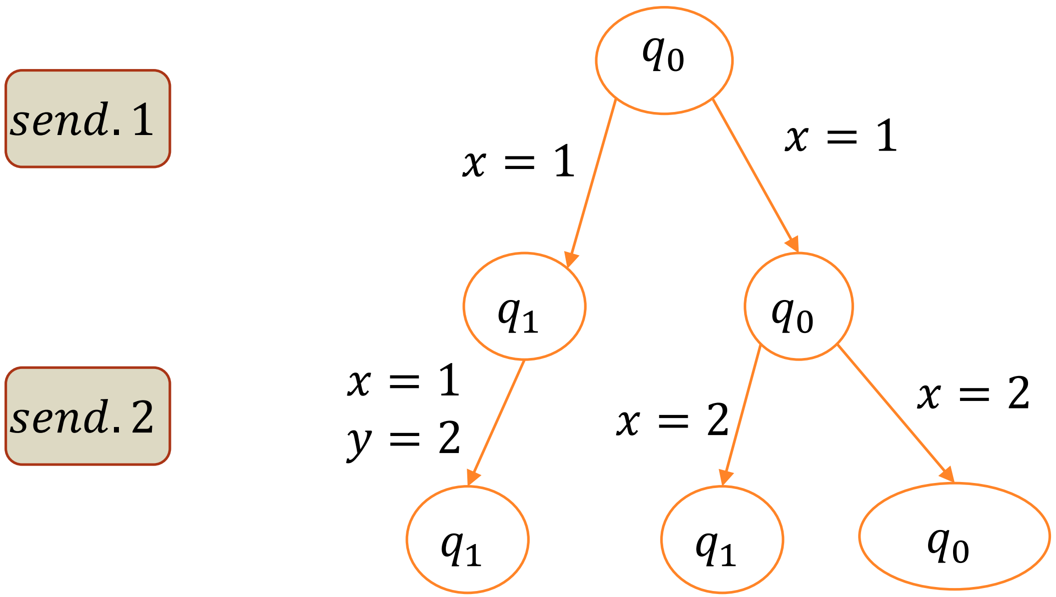
$send.1$

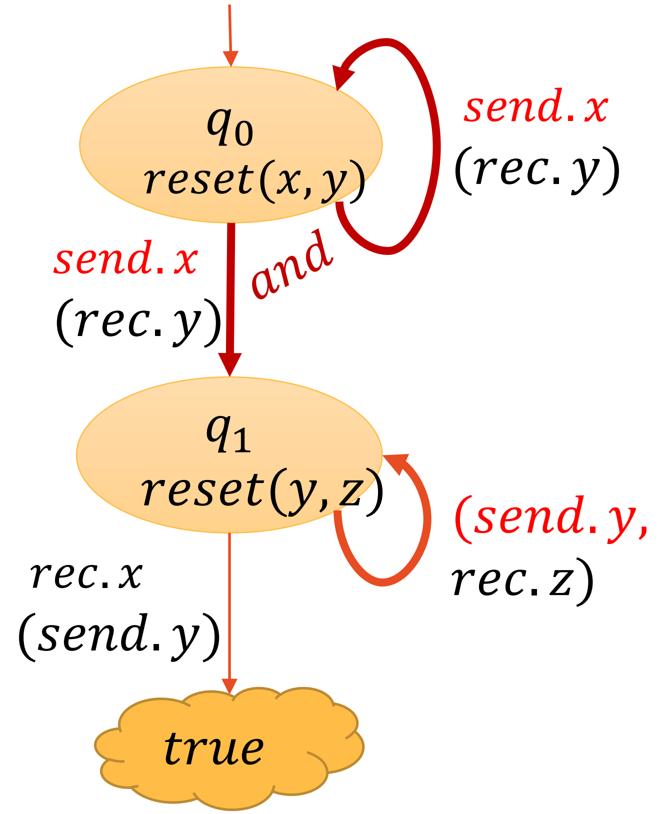




Alternating Variable Büchi Automata

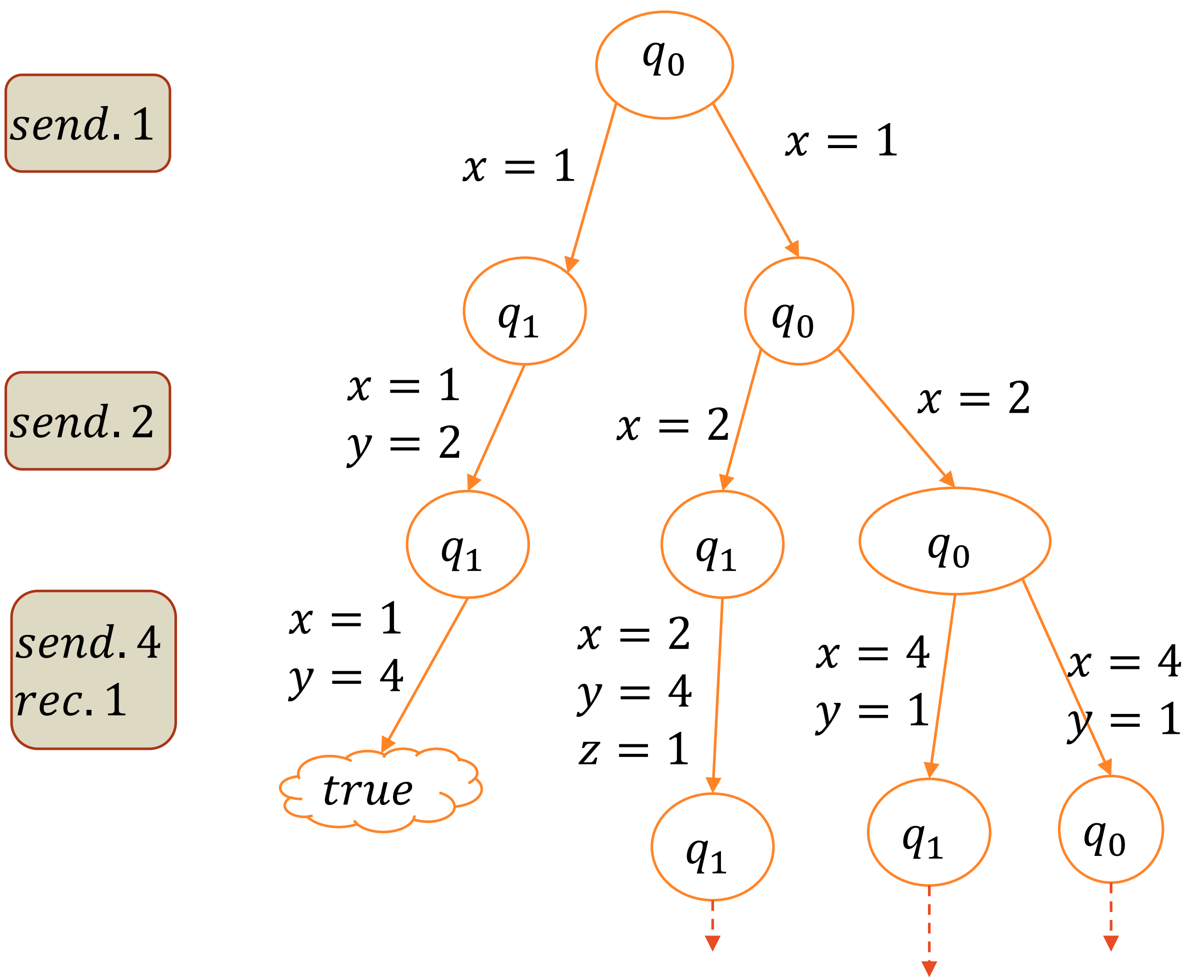
- $G\exists x(send.x \wedge XF\ rec.x)$

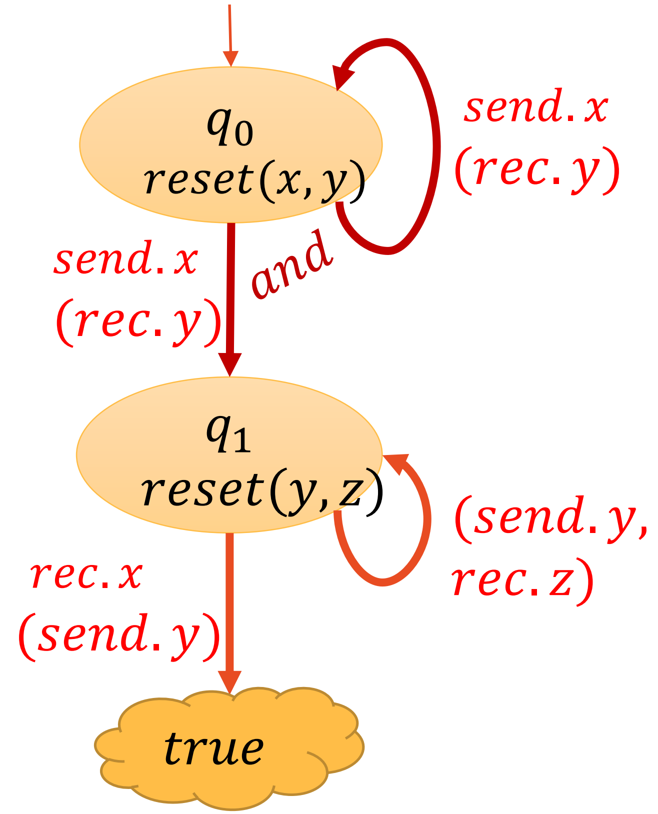




Alternating Variable Büchi Automata

- $G\exists x(send.x \wedge XF\, rec.x)$





Alternating Variable Automata are Stronger than Non-Det Variable Automata

- $G\exists x(send.x \wedge XF\ rec.x)$
- Consider the following computation

$send.1$ | $send.2$ $rec.1$ | $send.3$ | $send.4$ $rec.2$ | $send.5$ | $send.6$ $rec.3$ | $send.$

Unlike the finite alphabet case!

- In Büchi:
  Increasing gaps between $send.x, rec.x$.
  Not enough variables and states to remember all values

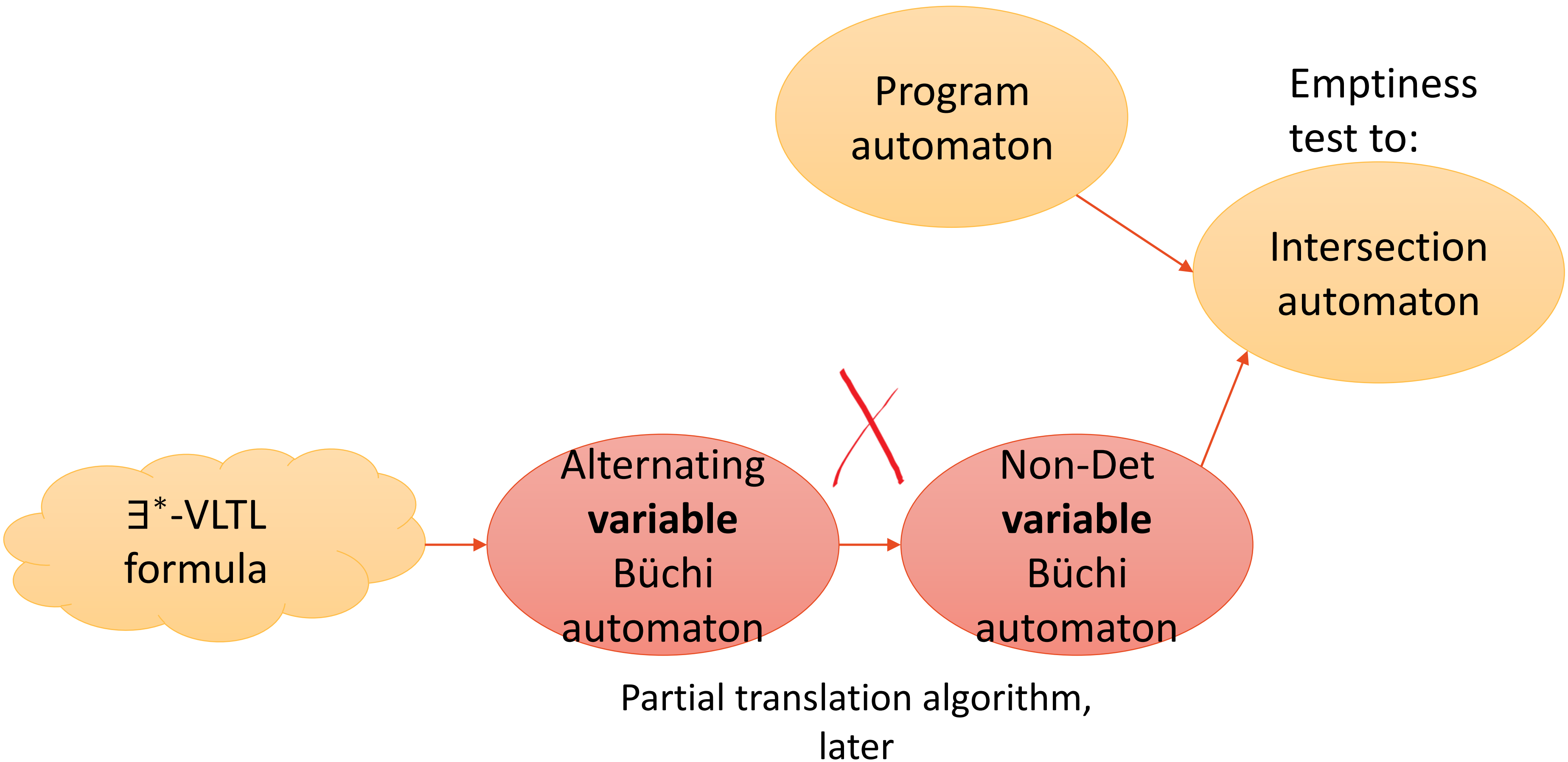
Alternating Variable Automata are Stronger than Non-Det Variable Automata

- $G\exists x(send.x \wedge XF\ rec.x)$
- Consider the following computation

$send.1$	$send.2$ $rec.1$	$send.3$	$send.4$ $rec.2$	$send.5$	$send.6$ $rec.3$	$send.7$	$send.8$ $rec.4$

- In Büchi:
  Increasing gaps between $send.x, rec.x$.
  Not enough variables and states to remember all values

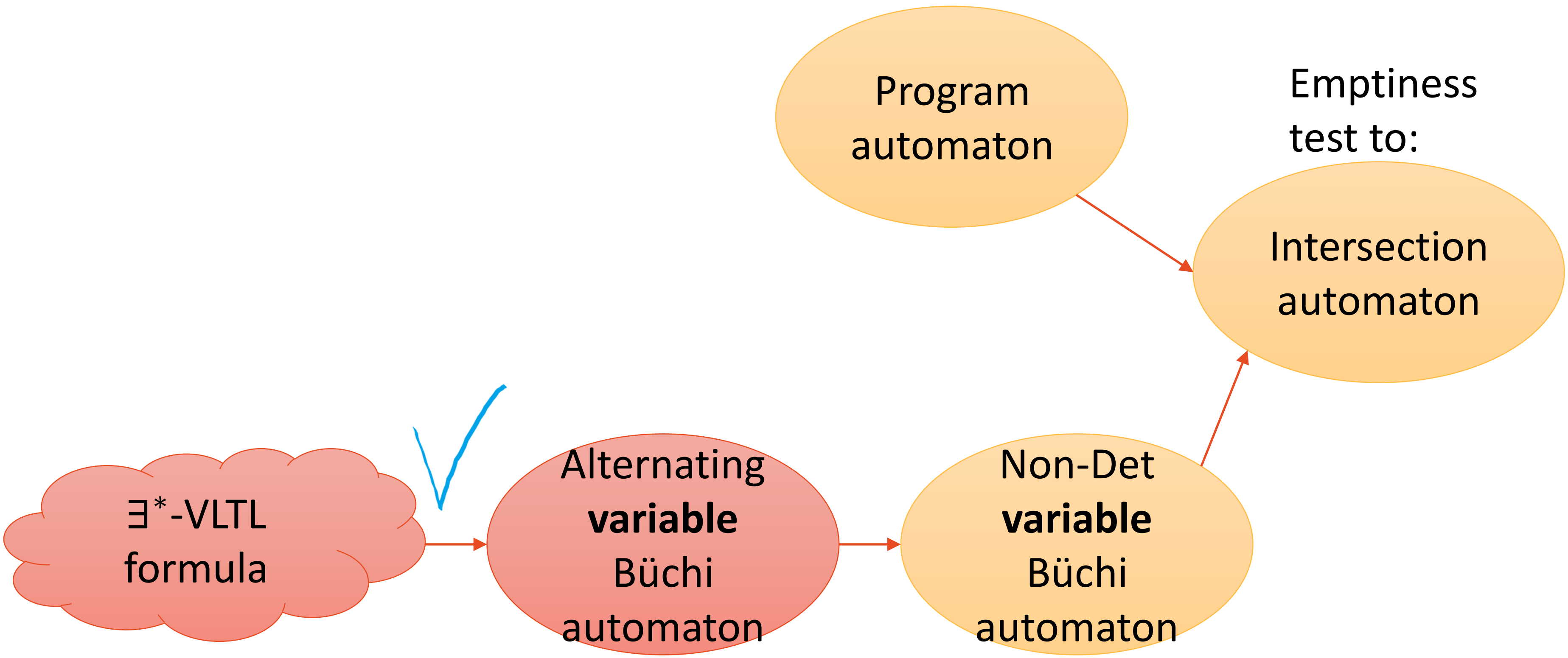
Model checking Process
Infinite Data Domains



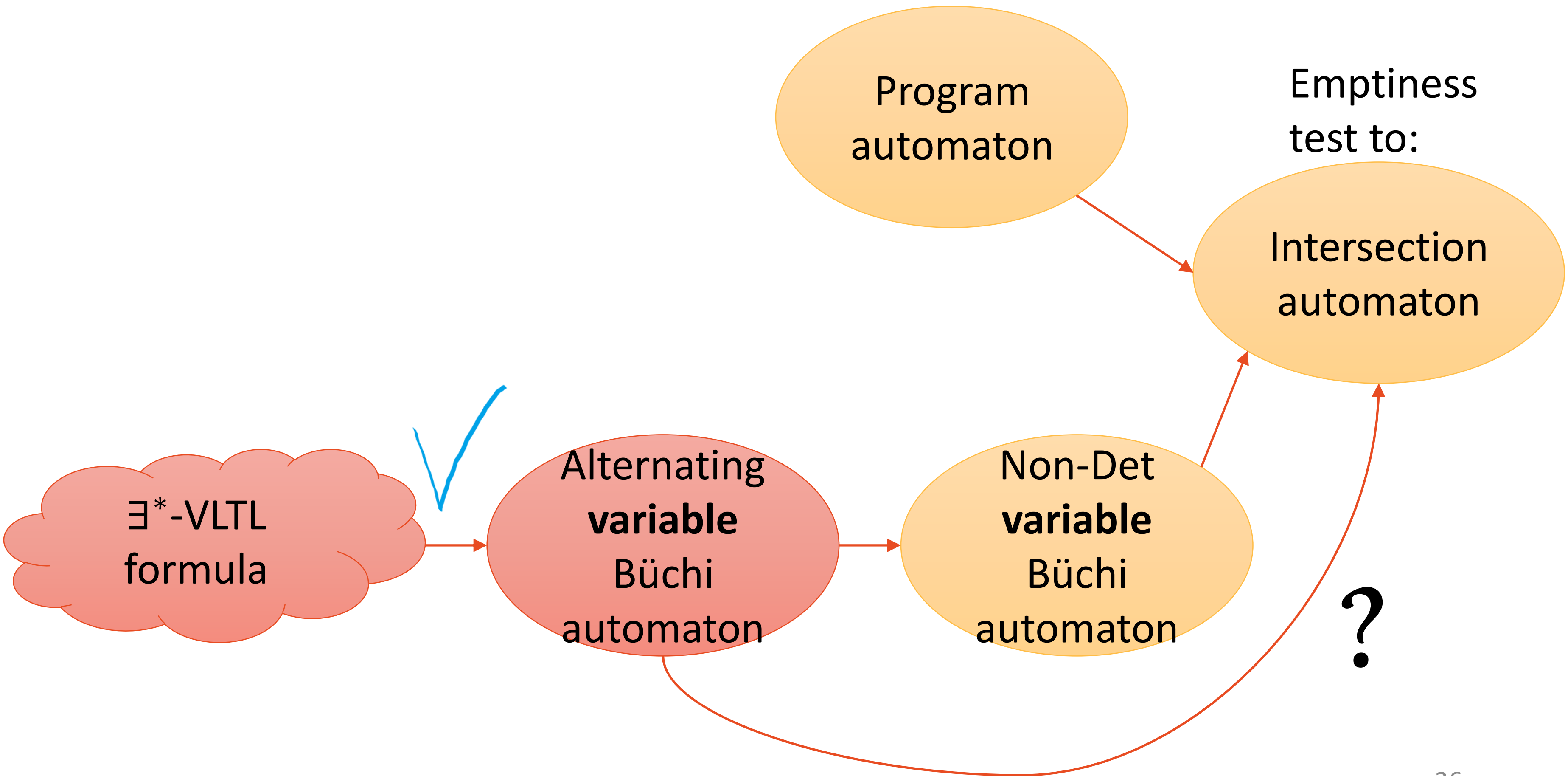
Alternating Variable Automata and $\exists^*$-VLTL

- All $\exists^*$-VLTL formulas can be translated into alternating variable automaton, based on [V96] construction
  - "$\exists x$" is translated to $reset(x)$ in the corresponding state

Model checking Process Infinite Data Domains



Model checking Process
Infinite Data Domains



Emptiness Test to Alternating Variable Büchi Automata?

- Not possible
- Satisfiability problem of $\exists^*$-VLTL formulas is undecidable [SW14]
- Emptiness problem $\equiv$ satisfiability problem
  - The formula automaton language is empty iff the formula is unsatisfiable

Recap - Alternating Variable Büchi Automata

- Can express all $\exists^*$-VLTL
- Are stronger than non-deterministic variable Büchi automata
- No emptiness test

Alternating Variable Büchi Automata → Non-Det Büchi Variable Automata

- A partial algorithm for translation
- Tracks paths that "owe" a visit in an accepting state [MH84]

  AND

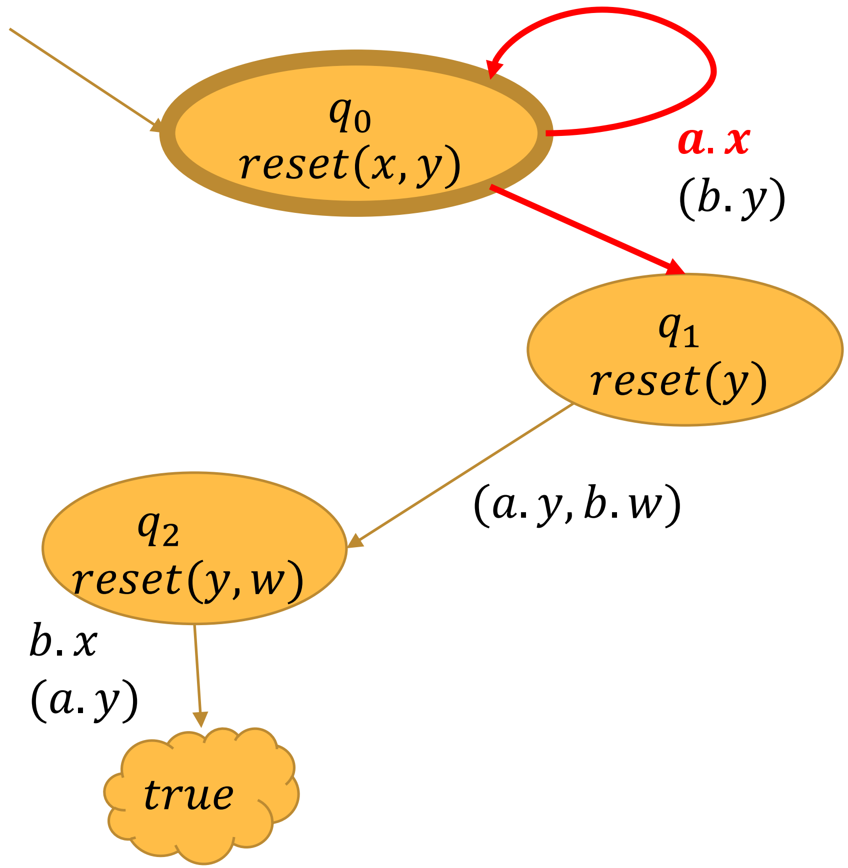
- Reuses variables that have been reset

Translation is possible when the number of new variables is bounded

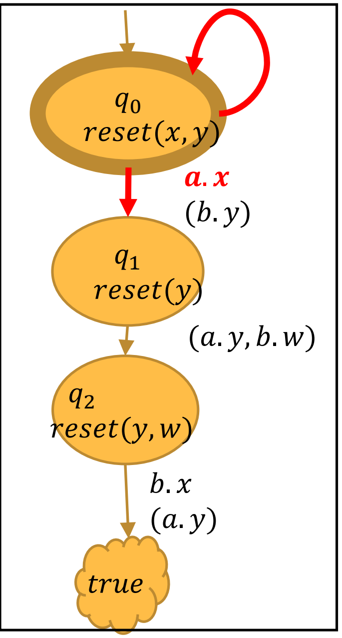
Translation is not possible when the number of new variables is unbounded

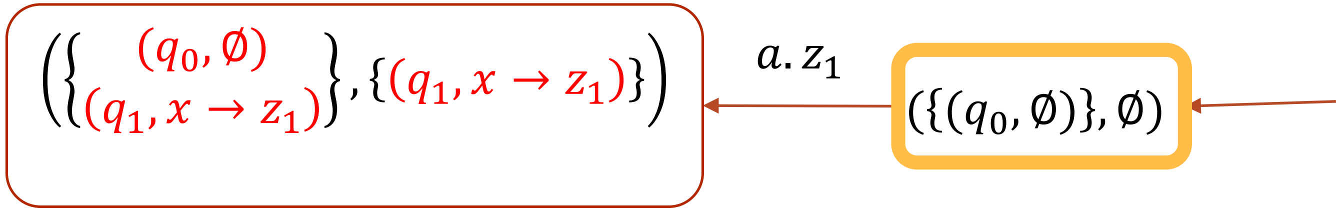
Example
Alternating Variable → Non-Det Variable

$G\exists x: a.x \wedge XX\ b.x$

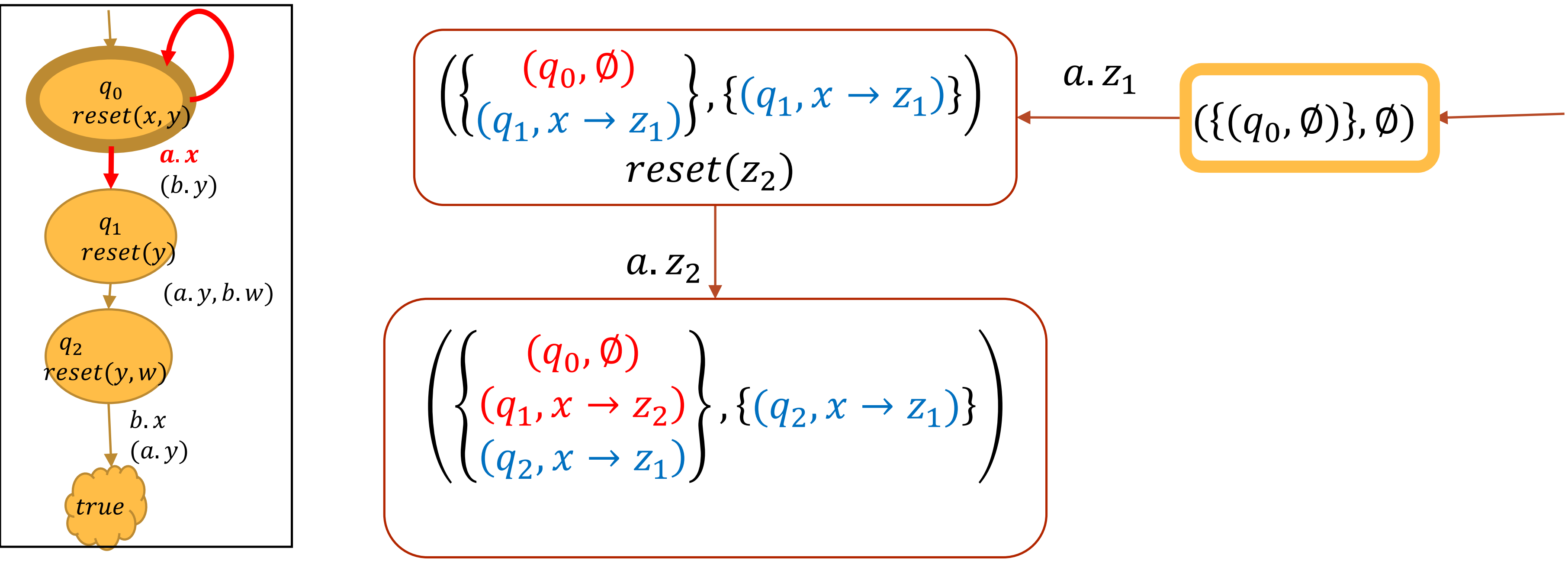


Example
Alternating Variable Automaton → Non-Det Variable Automaton

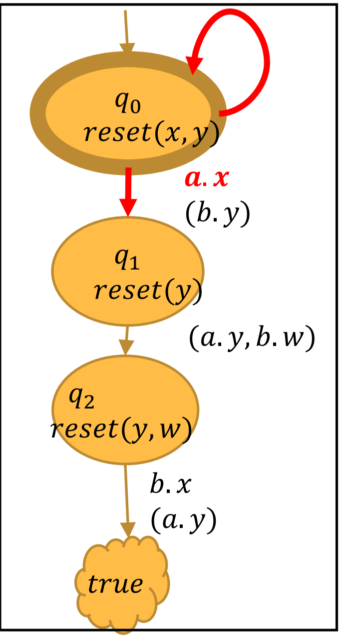


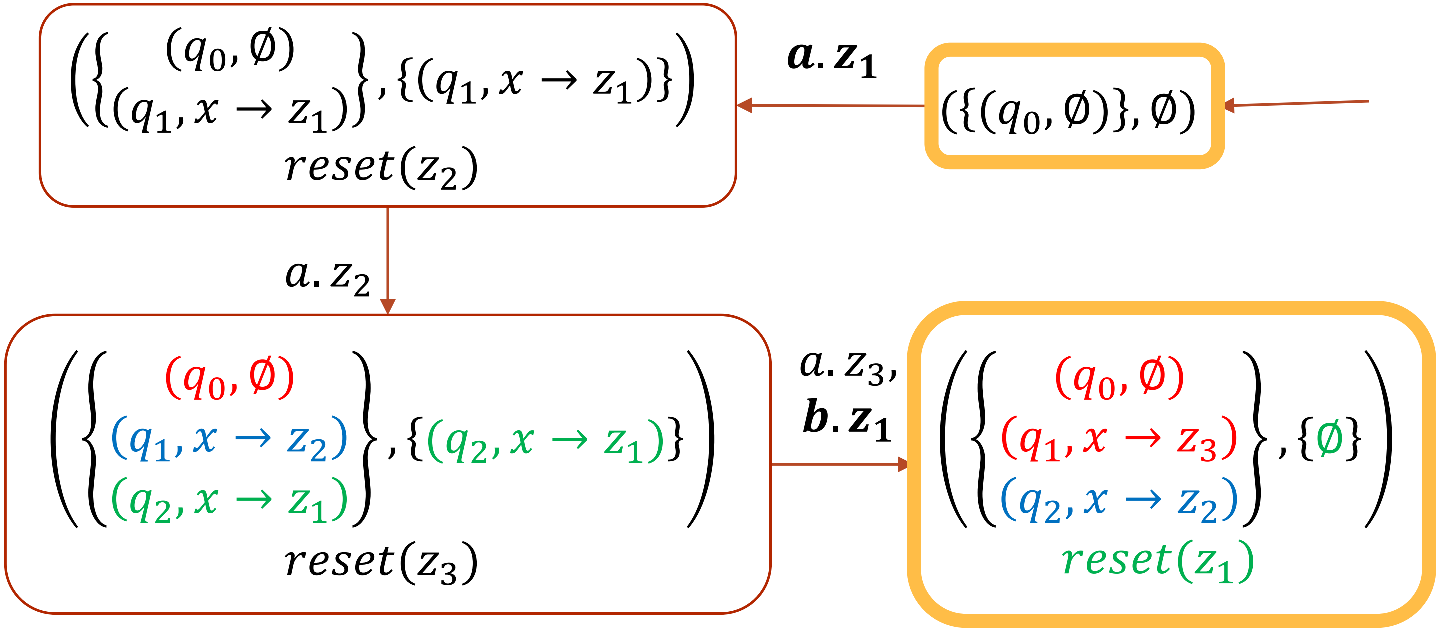


Example
Alternating Variable Automaton → Non-Det Variable Automaton

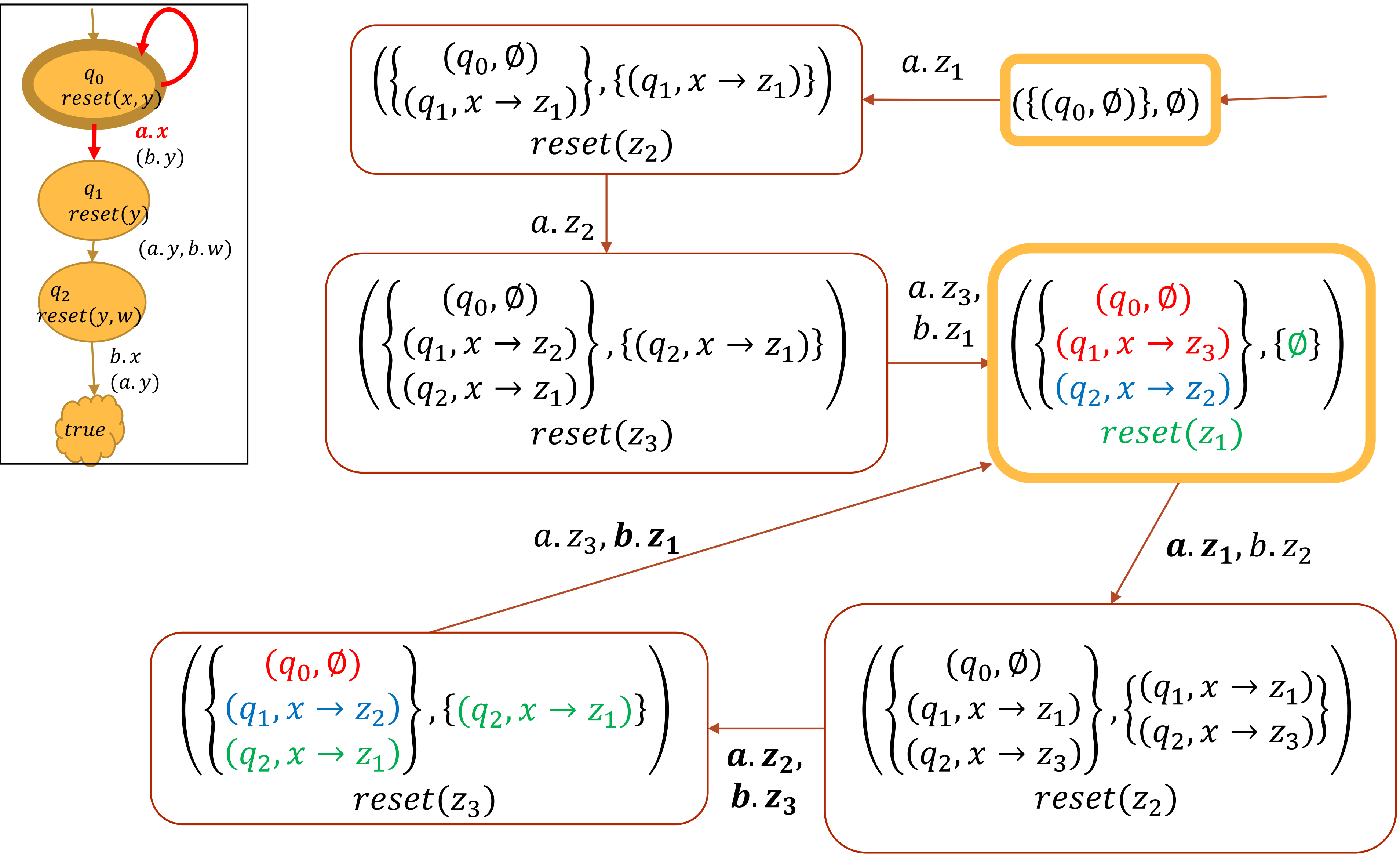


Example
Alternating Variable Automaton → Non-Det Variable Automaton



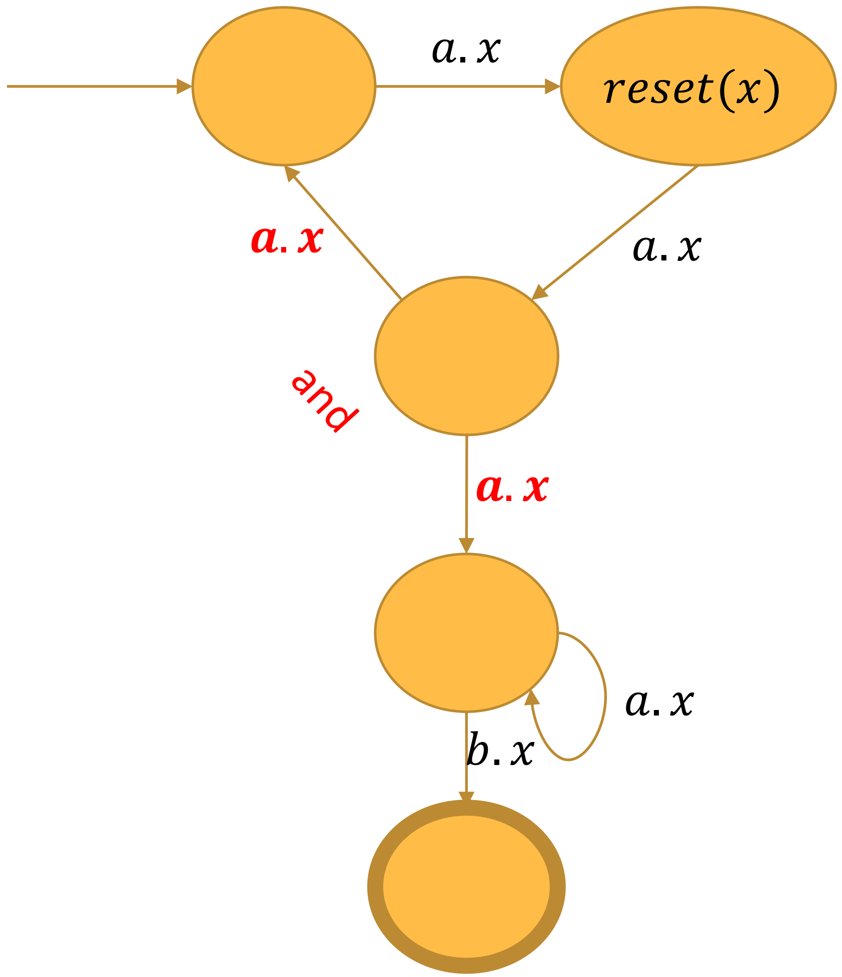


Example
Alternating Variable Automaton → Non-Det Variable Automaton



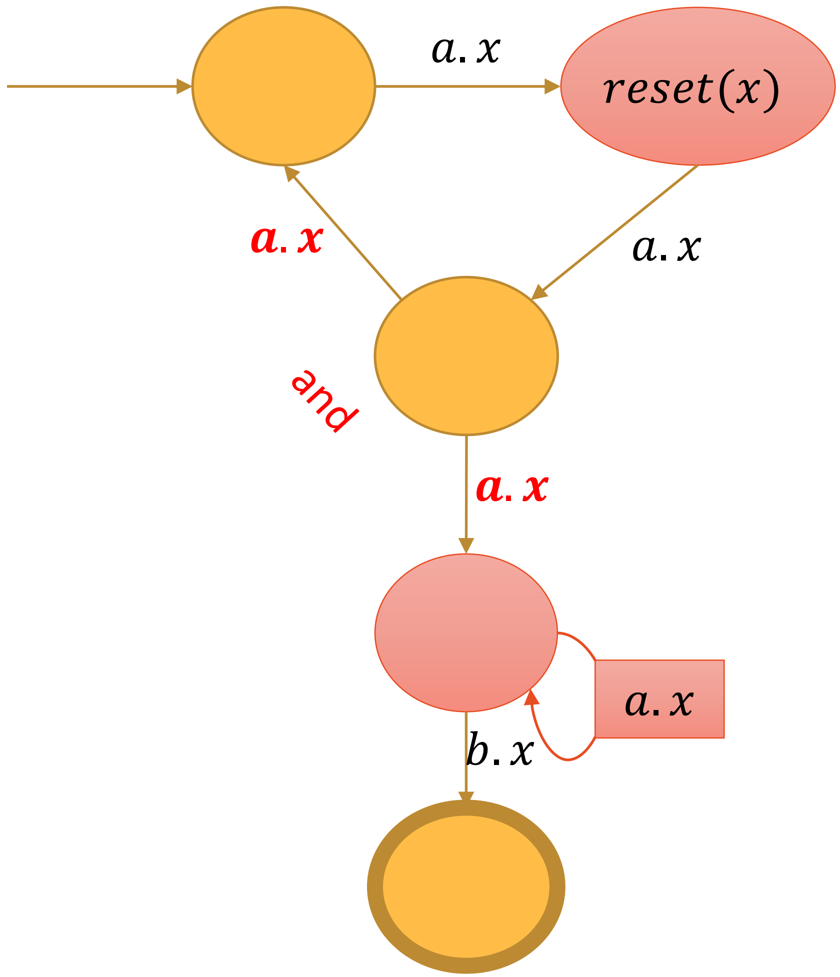
Example: **Translation fails**
Alternating Variable Automata → Non-Det Variable Automata

$$L = \emptyset$$



Example: **Translation fails**
Alternating Variable Automata → Non-Det Variable Automata

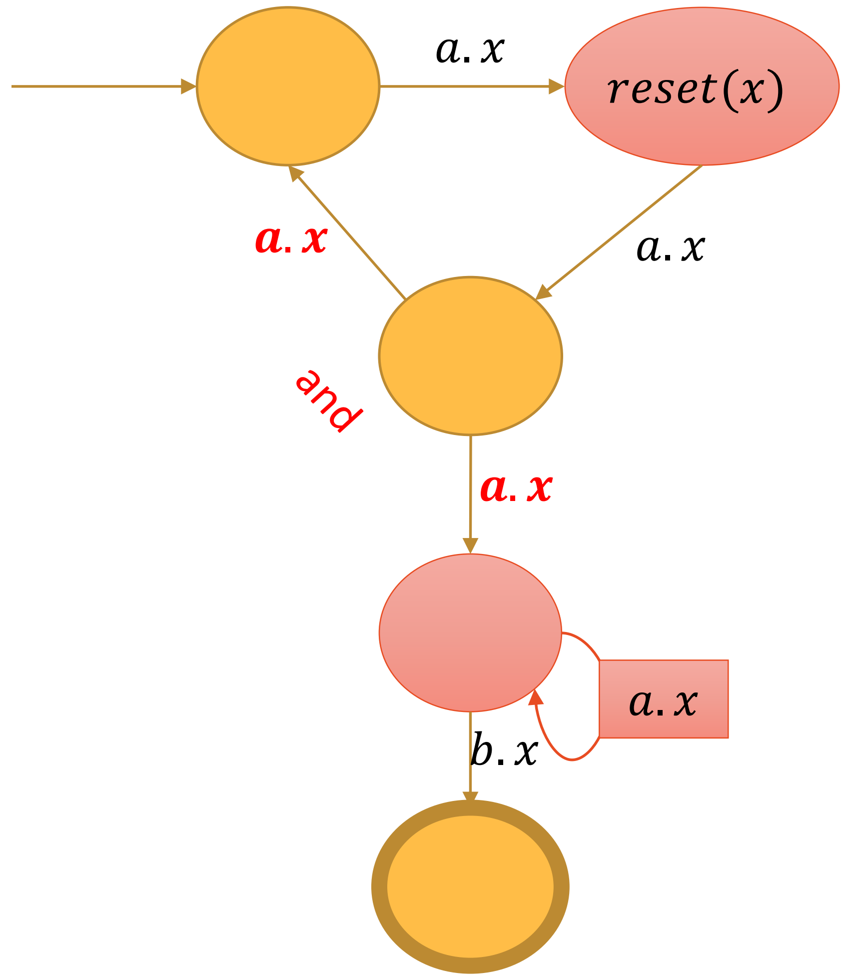
$$L = \emptyset$$



Example: **Translation fails**
Alternating Variable Automata → Non-Det Variable Automata

$$L = \emptyset$$

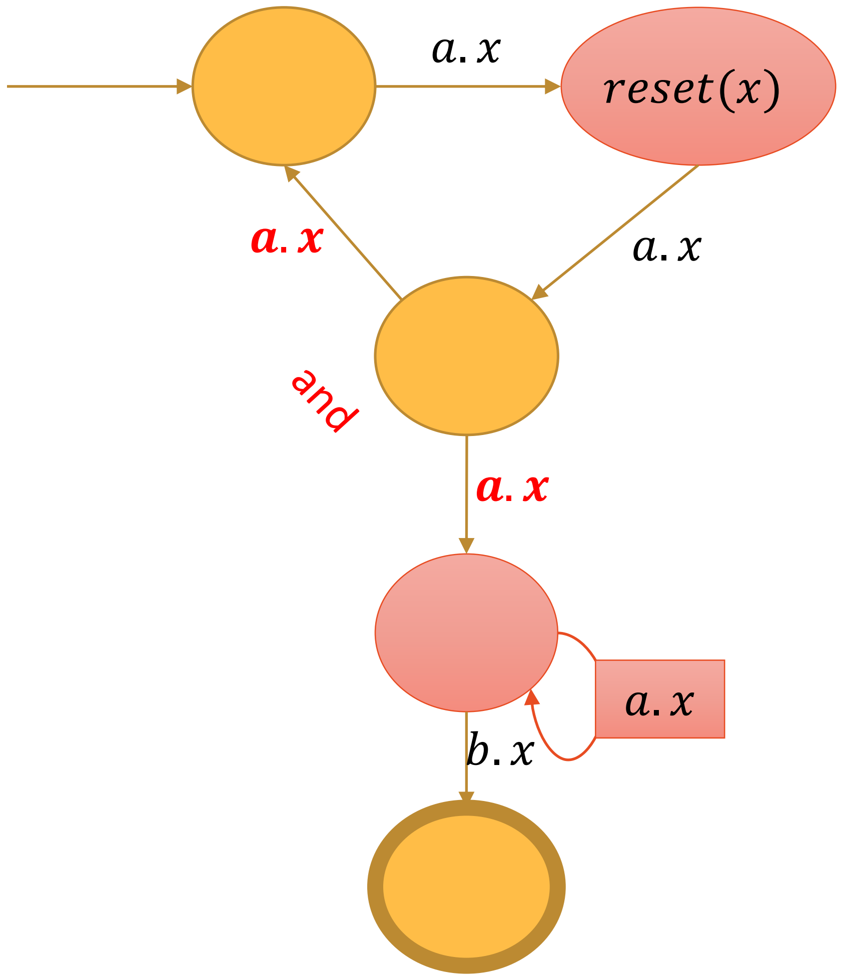
- Every translation algorithm is incomplete!



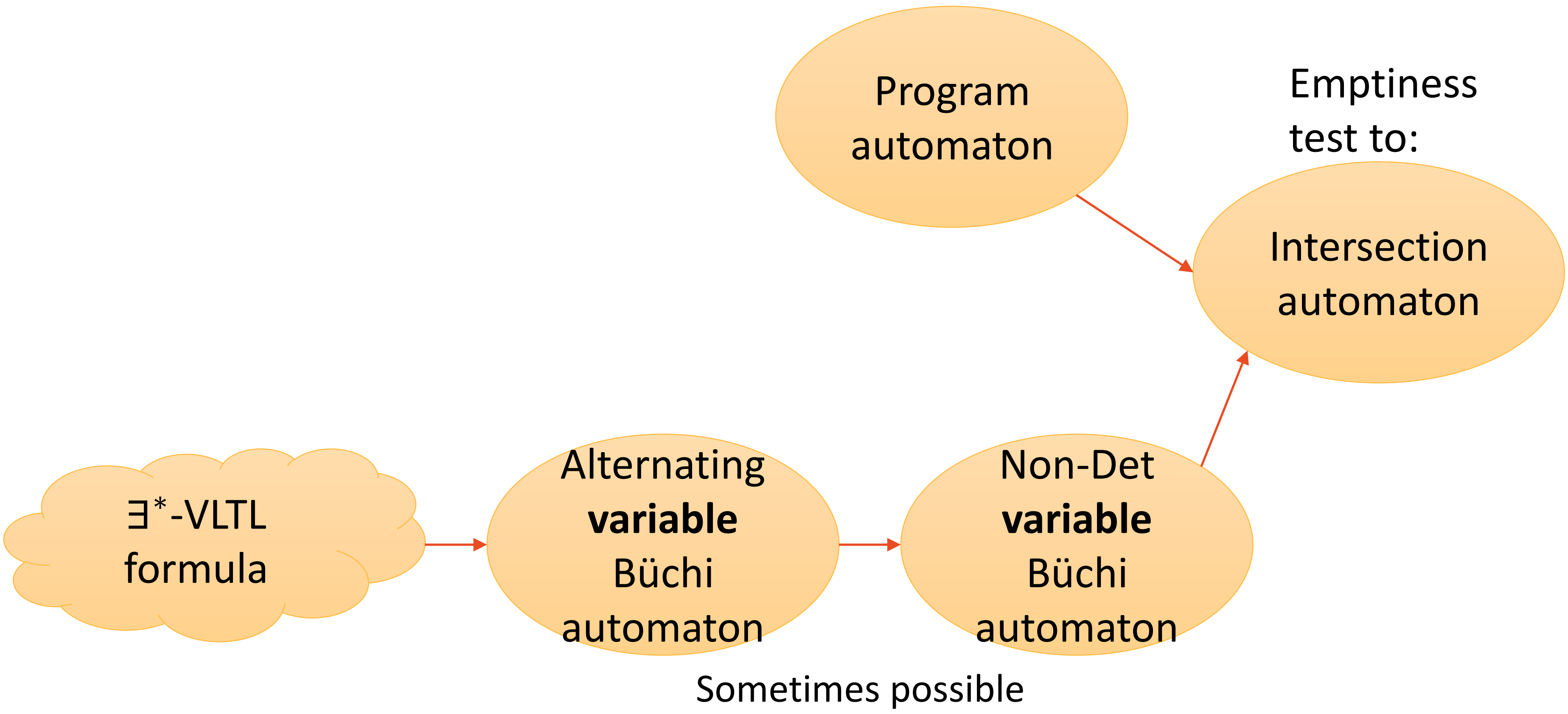
Example: **Translation fails**
Alternating Variable Automata → Non-Det Variable Automata

$$L = \emptyset$$

- Every translation algorithm is incomplete!
- Structural characterization for halting



Summary - Model checking Process Infinite Data Domains



Future work

- Bounded model checking algorithm for $\exists^*$-VLTL formulas, based on the partial translation algorithm.
- Extending our model to more expressive logics (Presburger / linear arithmetic)
- Synthesis

Questions?