Modeling Systems and Specifications over Infinite Data

Hadar Frenkel\textsuperscript{1}
Joint work with Orna Grumberg\textsuperscript{1} and Sarai Sheinvald\textsuperscript{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

YES!

NO! +
counter example

system

specification
Model checking

Given as an LTL formula

system

YES!

NO! + counter example

specification
LTL Specifications

• **Linear Temporal Logic** formulas.
• Describe occurrences of events during time

  • E.g.:

    \[ G(\text{send} \rightarrow F \text{receive}) \]

• Send, send, receive, send, receive,...
Model checking - automata theoretic approach

- System
- Specification
- YES!
- NO! + counter example
Model checking - automata theoretic approach

$A_P$
Recognizes all legal computations of the program

YES!
NO! + counter example

specification
Model checking - automata theoretic approach

$A_P$
Recognizes all legal computations of the program

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

YES!

NO! + counter example
Model checking - automata theoretic approach

\[ A_P \cap A_{\neg \varphi} \]

all computations of the program violating the specification

\[ A_P \]

“the program automaton”

\[ A_{\neg \varphi} \]

“the formula automaton”
Model checking - automata theoretic approach

- Program automaton
- Intersection automaton
- Emptiness test to: Formula automaton

LTL formula

Formula automaton
Model checking - automata theoretic approach

Reactive systems, Automata over infinite words

LTL formula → Alternating Büchi automaton → Non-Det Büchi automaton

Emptiness test to:
Intersection automaton

Program automaton
Model checking - automata theoretic approach

LTL formula

natural translation for LTL formulas

Program automaton

Emptiness test to:
Intersection automaton

Alternating Büchi automaton

Non-Det Büchi automaton
Model checking - automata theoretic approach

- LTL formula
- Alternating Büchi automaton
- Non-Det Büchi automaton
- Program automaton
- Intersection automaton
- Easy emptiness test
- Emptiness test to:

natural translation for LTL formulas
Non-Deterministic Büchi automaton Over Infinite Words

• $L = \{w \in \{a, b\}^\omega \mid w$ has infinite number of $a$’s $\}$

• Büchi automaton:

![Automaton Diagram]

• Visit an accepting state infinitely often
Alternating Büchi Automaton
Over Infinite Words

• Transition function is a positive boolean expression over states

$$\delta(q, a) = (q_0 \land q_1) \lor q_2$$

• 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

- $G (\text{send} \rightarrow \text{XF receive})$
Example - a Run of an Alternating Büchi automaton
Example - a Run of an Alternating Büchi automaton
Example - a Run of an Alternating Büchi automaton
Example - a Run of an Alternating Büchi automaton
Example - a Run of an Alternating Büchi automaton
Example - a Run of an Alternating Büchi automaton

Send
---
receive
Send
receive

q_0 → q_{0} → q_{0} → q_{0} → q_{0} → q_{1} → q_{1} → q_{1} → true

Receive/ ø
send
send/ ø
receive
true
Alternating to Non-Det \cite{MH84}

- Alternating Büchi automata are \textbf{equivalent} to non-deterministic Büchi automata
Alternating to Non-Det [MH84]

- Alternating Büchi automata are equivalent to non-deterministic Büchi automata
Alternating to Non-Det \cite{MH84}

- Alternating Büchi automata are equivalent to non-deterministic Büchi automata.
Alternating to Non-Det \[\text{[MH84]}\]

- Alternating Büchi automata are equivalent to non-deterministic Büchi automata.
Summary: Model checking - automata theoretic approach

- LTL formula
- Alternating Büchi automaton
- Program automaton
- Non-Det Büchi automaton
- Intersection automaton

Emptiness test to:
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:

\( \forall x: F \text{ active. } x \)

• \( AP \) - finite set of propositions

• \( V \) - finite set of variables

• Parameterized propositions instead of atomic propositions

• Quantifiers
$\exists^*\text{-VLTL}$

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

```
send. 1
send. 1
send. 4
send. 7
send. 3
send. 9
```
Model checking Process
Infinite Data Domains

∃*-VLTL formula

Program automaton

Intersection automaton

Emptiness test to:

Alternating Büchi automaton

Non-Det Büchi automaton
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

- $G \exists x: send. x$
Model checking Process
Infinite Data Domains

∃*-VLTL formula

Alternating variable Büchi automaton

Program automaton

Non-Det variable Büchi automaton

Emptiness test to:

Intersection automaton
Alternating Büchi Variable Automata [FGS17]

• Similar semantics as non-deterministic

\[ G \exists x (\text{send}. \, x \land X F \, \text{rec}. \, x) \]
Alternating Variable Büchi Automata

• $G \exists x (send. x \land XF rec. x)$

send. 1

$q_0$

$x = 1$

$q_1$

$q_0$

$x = 1$

$q_0$

reset($x, y$)

$q_0$

$send. x$

(rec. $y$)

$q_1$

reset($y, z$)

$q_1$

(send. $y$, rec. $z$)

true
Alternating Variable Büchi Automata

- $G \exists x (\text{send}. \, x \land XF \, \text{rec}. \, x)$
Alternating Variable Büchi Automata

• $G \exists x (send. x \land XF \text{ rec. } x)$
Alternating Variable Automata are Stronger than Non-Det Variable Automata

• $G \exists x (send. x \land XF rec. x)$

• Consider the following computation

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

Unlike the finite alphabet case!
Alternating Variable Automata are Stronger than Non-Det Variable Automata

• $G \exists x (send. x \land XF rec. x)$

• Consider the following computation

• In Büchi:
  Increasing gaps between $send. x, rec. x$.
  Not enough variables and states to remember all values
Model checking Process
Infinite Data Domains

∃*-VLTL formula

Program automaton

Intersection automaton

Emptiness test to:

Alternating variable Büchi automaton

Non-Det variable Büchi automaton

Partial translation algorithm, later
Alternating Variable Automata and $\exists^*\text{-VLTL}$

- All $\exists^*\text{-VLTL}$ formulas can be translated into alternating variable automaton, based on [V96] construction
  - "$\exists x$" is translated to $\text{reset}(x)$ in the corresponding state
Model checking Process
Infinite Data Domains

∃*-VLTL formula

Alternating variable Büchi automaton

Program automaton

Non-Det variable Büchi automaton

Emptiness test to:

Intersection automaton
Model checking Process
Infinite Data Domains

∃*-VLTL formula

Alternating variable Büchi automaton

Program automaton

Intersection automaton

Emptiness test to:

Non-Det variable Büchi automaton
Emptiness Test to Alternating Variable Büchi Automata?

• Not possible

• Satisfiability problem of $\exists^*\text{-VLTL}$ 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 $\exists^*\text{-VLTL}$
• Are stronger than non-deterministic variable Büchi automata
• No emptiness test
Alternating Variable Büchi Automata $\rightarrow$ Non-Det Büchi Variable Automata

- A partial algorithm for translation
- Tracks paths that “owe” a visit in an accepting state \cite{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

$G \exists x : a \cdot x \land XX b \cdot x$

Diagram:

- $q_0$ reset($x, y$)
- $q_1$ reset($y$)
- $q_2$ reset($y, w$)

Transitions:
- $a \cdot x (b \cdot y)$ from $q_0$ to $q_1$
- $b \cdot x (a \cdot y)$ from $q_2$ to $true$
- $(a \cdot y, b \cdot w)$ from $q_2$ to $q_1$

Initial state: $q_0$

Final state: $true$
Example
Alternating Variable Automaton $\rightarrow$ Non-Det Variable Automaton

$$
\begin{align*}
&\{ (q_0, \emptyset) \}, \{ (q_1, x \rightarrow z_1) \} \\
&\{ (q_0, \emptyset) \}, \emptyset \\
\end{align*}
$$
Example
Alternating Variable Automaton $\rightarrow$ Non-Det Variable Automaton

\[
\begin{align*}
&\left\{ (q_0, \emptyset) \right\}, \left\{ (q_1, x \rightarrow z_1) \right\} \\
&\text{reset}(z_2)
\end{align*}
\]

\[
\begin{align*}
&\left\{ (q_0, \emptyset) \right\}, \left\{ (q_1, x \rightarrow z_2) \right\}, \left\{ (q_2, x \rightarrow z_1) \right\} \\
&\text{reset}(z_2)
\end{align*}
\]
Example
Alternating Variable Automaton $\rightarrow$ Non-Det Variable Automaton

$$
\begin{align*}
&\text{reset}(x, y) \\
&\text{reset}(y, w) \\
&\text{true}
\end{align*}
$$

\begin{align*}
(q_0, \emptyset), (q_1, x \rightarrow z_1), (q_2, x \rightarrow z_1), (q_0, \emptyset) \\
\text{reset}(z_2), \text{reset}(z_3), \text{reset}(z_1)
\end{align*}
Example
Alternating Variable Automaton $\rightarrow$ Non-Det Variable Automaton

\[ (\{ (q_0, \emptyset) \}, \{(q_1, x \rightarrow z_1)\}) \]

- reset(z_2)

\[ (\{ (q_0, \emptyset) \}, \{(q_1, x \rightarrow z_2)\}) \]

- reset(z_3)

\[ (\{ (q_0, \emptyset) \}, \{(q_1, x \rightarrow z_3)\}) \]

- reset(z_1)

\[ (\{ (q_0, \emptyset) \}, \{(q_1, x \rightarrow z_1)\}) \]

- reset(z_2)

\[ (\{ (q_0, \emptyset) \}, \{(q_1, x \rightarrow z_3)\}) \]

- reset(z_2)
Example: Translation fails
Alternating Variable Automata $\rightarrow$ Non-Det Variable Automata

$L = \emptyset$
Example: Translation fails
Alternating Variable Automata $\rightarrow$ 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

∃*-VLTL formula

Alternating variable Büchi automaton

Sometimes possible

Program automaton

Intersection automaton

Emptiness test to:

Non-Det variable Büchi automaton
Future work

• Bounded model checking algorithm for $\exists^*-\text{VLTL}$ formulas, based on the partial translation algorithm.

• Extending our model to more expressive logics (Presburger / linear arithmetic)

• Synthesis
Questions?