UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Course

Runtime Verification

Martin Leucker (ISP)
Volker Stolz (Høgskolen i Bergen, NO)

INF5140 / V17

Chapters of the Course

Chapter 1 Recall Runtime Verification in More Depth
Chapter 2 Specification Languages on Words
Chapter 3 LTL on Finite Words
Chapter 4 Impartial Runtime Verification
Chapter 5 Anticipatory LTL Semantics
Chapter 6 Alternating Büchi Automata
Chapter 7 Monitor Construction for Anticipatory Runtime Verification
Chapter 8 LTL with a Predictive Semantics
Chapter 9 Runtime Verification Summary

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Chapter 1

Recall Runtime Verification in More Depth

Course "Runtime Verification"

M. Leucker & V. Stolz

INF5140 / V17

Chapter 1

Learning Targets of Chapter "Recall Runtime Verification in More Depth".

1. Recall the underlying principle of runtime verification.
2. Get to know to applications of runtime verification.
3. See different frameworks for runtime verification.

Chapter 1

Outline of Chapter "Recall Runtime Verification in More Depth".

Runtime Verification

- Recall
- Word Problem
- Good Monitors?

Applications

- Runtime Reflection
- When to Use RV?
- RV Frameworks

Section

Runtime Verification

Recall
Word Problem
Good Monitors?

Chapter 1 “Recall Runtime Verification in More Depth”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

Runtime Verification (Recall)

Verification technique that allow for checking whether a run of a system under scrutiny satisfies or violates a given correctness property.

Targets & Outline

Runtime Verification
- Recall
- Word Problem
- Good Monitors?

Applications
- Runtime Reflection
- When to Use RV?
- RV Frameworks

Conclusion

Run and Execution (Recall)

- Run: possibly infinite sequence of the system's states.
- Run formally: possibly infinite word or trace.

Targets & Outline

Runtime Verification
Recall
Word Problem
Good Monitors?

Applications
Runtime Reflection
When to Use RV?
RV Frameworks

Conclusion

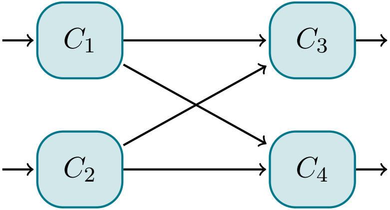
Run and Execution (Recall)

- Run: possibly infinite sequence of the system's states.
- Run formally: possibly infinite word or trace.

- Execution: finite prefix of a run.
- Execution formally: finite word or trace.
- RV is primarly used on executions.

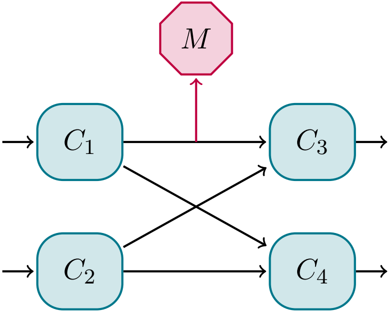
Adding Monitors to a System (Recall)

- A monitor checks whether an execution meets a correctness property.
- A monitor is a device that reads a finite trace and yields a certain verdict.



Adding Monitors to a System (Recall)

- A monitor checks whether an execution meets a correctness property.
- A monitor is a device that reads a finite trace and yields a certain verdict.



Targets & Outline

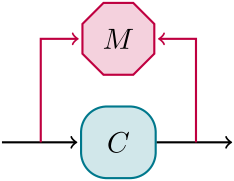
Runtime Verification
- Recall
- Word Problem
- Good Monitors?

Applications
- Runtime Reflection
- When to Use RV?
- RV Frameworks

Conclusion

Monitors can Check Relations of Values (Recall)

- A monitor can use more than one input value.
- A monitor can check the relations of multiple values.



RV and the Word Problem

A simple monitor outputs

- yes if the execution satisfies the correctness property,
- no if not.

- Let $[\![\varphi]\!]$ denote the set of valid executions given by property φ.
- Then runtime verification answers the word problem $w \in [\![\varphi]\!]$.
- The word problem can be decided with lower complexity compared to the subset problem.

Targets & Outline

Runtime Verification
Recall
Word Problem
Good Monitors?

Applications
Runtime Reflection
When to Use RV?
RV Frameworks

Conclusion

How Does a *Good Monitor* Look?

Impartiality and *Anticipation*

Definition (Impartiality)

Impartiality requires that a finite trace is not evaluated to $true$ or, respectively $false$, if there still exists a (possibly infinite) continuation leading to another verdict.

Definition (Anticipation)

Anticipation requires that once every (possibly infinite) continuation of a finite trace leads to the same verdict, then the finite trace evaluates to this very same verdict.

A monitor for RV should adhere to both maxims!

Section

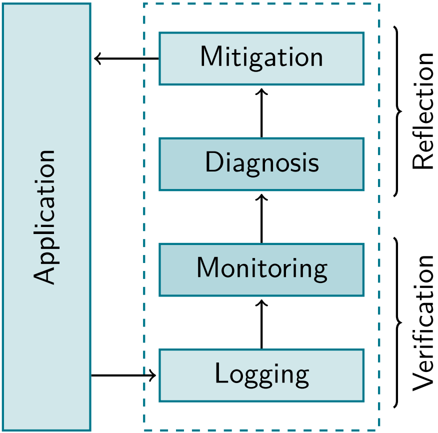
Applications

Runtime Reflection
When to Use RV?
RV Frameworks

Runtime Reflection

Runtime reflection (RR) is an architecture pattern for the development of reliable systems.

- A *monitoring layer* is enriched with
- a *diagnosis layer* and a subsequent
- *mitigation layer*.



Runtime Reflection

Logging—Recording of System Events

The logging layer

- observes system events and
- provides them for the monitoring layer.

Realization

- Add code annotations within the system to build or
- use separated stand-alone loggers.

Runtime Reflection

Monitoring—Fault Detection

The monitoring layer

- is implemented using runtime verification techniques,
- consists of a number of monitors,
- detects the presence of faults in the system and
- raises an alarm for the diagnosis layer in case of faults.

Targets & Outline

Runtime Verification
Recall
Word Problem
Good Monitors?

Applications
Runtime Reflection
When to Use RV?
RV Frameworks

Conclusion

Runtime Reflection

Diagnosis—Failure Identification

The diagnosis layer

- collects the verdicts of the monitors and
- deduces an explanation for the current system state solely based upon the results of the monitors and general information on the system.

Runtime Reflection

Mitigation—Reconfiguration

The reconfiguration layer

- mitigates the failure, if possible,
- or else may store detailed diagnosis information for off-line treatment.

When to Use RV?

- The verification verdict is often referring to a model of the real system. Runtime verification may then be used to easily check the actual execution of the system. Thus, runtime verification may act as a partner to theorem proving and model checking.
- Often, some information is available only at runtime. In such cases, runtime verification is an alternative to theorem proving and model checking.
- The behavior of an application may depend heavily on the environment of the target system. In this scenario, runtime verification adds on formal correctness proofs by model checking and theorem proving.
- In the case of systems where security is important, it is useful also to monitor behavior or properties that have been statically proved or tested.

Targets & Outline

Runtime Verification
Recall
Word Problem
Good Monitors?

Applications
Runtime Reflection
When to Use RV?
RV Frameworks

Conclusion

Taxonomy

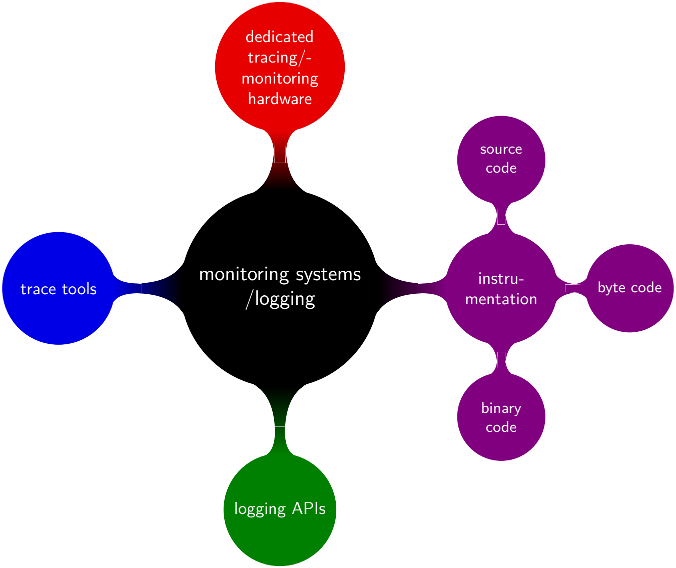
Targets & Outline

Runtime Verification
Recall
Word Problem
Good Monitors?

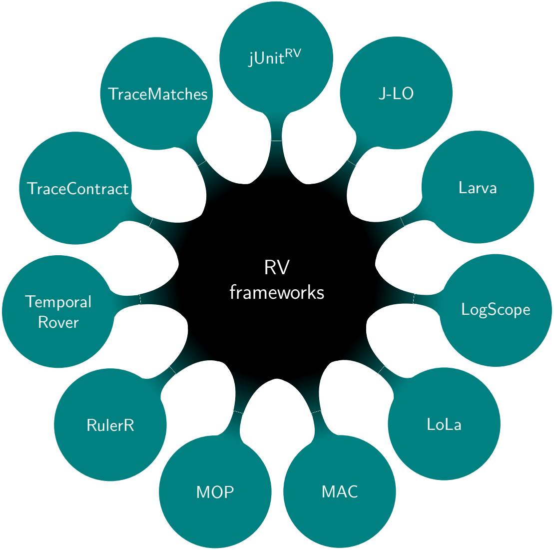
Applications
Runtime Reflection
When to Use RV?
RV Frameworks

Conclusion

Monitoring Systems/Logging: Overview



Monitoring Systems/Logging: Overview



Conclusion

1. Runtime verification deals with verification techniques that allow checking whether an execution of a system under scrutiny satisfies or violates a given correctness property.
2. A Monitor checks whether an execution meets a correctness property.
3. One of its main technical challenges is the synthesis of efficient monitors from logical specifications.

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING AND PROGRAMMING LANGUAGES

Chapter 2

Specification Languages on Words

Course "Runtime Verification"
M. Leucker & V. Stolz
INF5140 / V17

Chapter 2

Learning Targets of Chapter "Specification Languages on Words".

1. Understand that RV specifies shape of words.
2. Recall the idea of regular expressions and understand their limitations for practical specifications.
3. Get an idea about temporal logics.
4. Understand the difference of regular expressions and temporal logics.
5. Understand how to specify properties in LTL.

Chapter 2

Outline of Chapter "Specification Languages on Words".

Runs Are Words

- States of the System
- Executions Are Words

Regular Expressions

- The Idea
- Syntax and Semantics
- Limitations

Linear Temporal Logic (LTL)

- Propositional Logic
- Temporal Logic

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Section

Runs Are Words

States of the System
Executions Are Words

Chapter 2 “Specification Languages on Words”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

Recap

We want to monitor the execution of a system.

We have already seen that

- A run of a system is a possibly infinite sequence of the system's states.
- An execution of a system is a finite prefix of a run.

Observations

- We describe the execution of a system in a discrete way.
- The system is in exactly one state at a time.
- In the next step the system is in the next state.

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Atomic Propositions

- An atomic proposition is an indivisible bit.
- We consider a fixed set of finitely many such bits.
- In every state every atomic proposition is either true or false.
- In other words:
  In every state of the execution some atomic propositions hold.

Example

- Variable `count` is greater than 5.
- Memory for a variable `data` is allocated.
- Memory for `data` is free.
- The file handle `logfile` points to an opened file.

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

States

- Let AP be a fixed finite non empty set of atomic propositions.
- $\Sigma = 2^{\mathrm{AP}}$ is the power set of these.
- A state can be seen as an element $a \in \Sigma$.

Targets & Outline

Runs Are Words

States of the System

Executions Are Words

Regular Expressions

The Idea

Syntax and Semantics

Limitations

Linear Temporal Logic (LTL)

Propositional Logic

Temporal Logic

Conclusion

Executions Are Like Linear Paths

Languages over Alphabets

Let Σ be an alphabet and $n \in \mathbb{N}$.
We then use the following notation:

Notation	Meaning
Σ^*	set of all finite words over Σ
Σ^n	all words in Σ^* of length n
$\Sigma^{\leq n}$	all words in Σ^* of length at most n
$\Sigma^{\geq n}$	all words in Σ^* of length at least n
Σ^+	$= \Sigma^{\geq 1}$
Σ^ω	set of all infinite words over Σ
Σ^∞	$= \Sigma^* \cup \Sigma^\omega$

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Executions Are Words

- A state can be seen as an element $a \in \Sigma$.
- Now a run is an infinite word $w \in \Sigma^\omega$
- and an execution a finite prefix $w \in \Sigma^*$.

Runtime verification is about checking if an execution is correct, so we need to specify the set of correct executions as a language $L \subseteq \Sigma^*$. Therefore a correctness property is a language L.

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Section

Regular Expressions

The Idea
Syntax and Semantics
Limitations

Chapter 2 “Specification Languages on Words”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

Regular Expressions: The Idea

- Use a bottom up construction to construct a complex language by combining simpler languages together.
- Start with languages containing only one word of length 1.
- Use the common operations on languages to combine these into complexer languages.

Operations on Languages

Let $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ be two languages. We than have

intersection $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \wedge w \in L_2\}$

union $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \vee w \in L_2\}$

complement $\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$

concatenation $L_1 \circ L_2 = \{uv \in \Sigma^* \mid u \in L_1 \wedge v \in L_2\}$

Kleene star $L^* = \{u_1 u_2 \ldots u_n \in \Sigma^* \mid \forall i : u_i \in L\}$

Regular Expressions

Regular expressions use only the operations union, concatenation and Kleene star. These are enough to build all regular languages.

- Every symbol $a \in \Sigma$ is a regular expression.
- The empty word ε describes the empty word.
- Concatenation is expressed by concatenating regular expressions.
- Union is expressed by the $\mid$ operator combining two regular expressions.
- Kleene star is expressed by the $\star$ operator at the end of a regular expression.

Examples

Examples

Let $\Sigma = \{0, 1\}$ be the finite alphabet.

- `(0|1)*` specifies all words $w \in \Sigma^*$.
- `1*0*` specifies all words $w \in \Sigma^*$ that do not contain the string `01`.
- `((0|1)1)*` specifies all words $w \in \Sigma^*$ of even length where every second letter is `1`.
- `((0|1)1)*(0|1|`ε`)` specifies all words $w \in \Sigma^*$ where every second letter is `1`.

Syntax of Regular Expressions

Definition (Syntax of regular expressions)

Let $x \in \Sigma$ be a symbol from a given alphabet. The syntax of regular expressions is inductively defined by the following grammar:

$$\varphi ::= \varepsilon \mid x \mid \varphi\varphi \mid (\varphi \,|\, \varphi) \mid (\varphi)\star$$

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Semantics of Regular Expression

Definition (Semantics of Regular Expressions)

Let $w, u_i \in \Sigma^*$ be words over the given alphabet, $x \in \Sigma$ be an element of the alphabet and R, R' regular expressions. Then the semantics of a regular expression is inductively defined as relation $\models$ of a non empty word and a regular expression as follows.

$$
\begin{aligned}
\varepsilon &\models \varepsilon \\
x &\models x \\
w &\models RR' && \text{iff } \exists u_1, u_2 : w = u_1 u_2 \\
& && \text{and } u_1 \models R \text{ and } u_2 \models R' \\
w &\models (R \mid R') && \text{iff } w \models R \text{ or } w \models R' \\
w &\models (R)\star && \text{iff } \exists u_1, \ldots, u_n : w = u_1 \ldots u_n \\
& && \text{and } \forall i \in \{1, \ldots, n\} : u_i \models R.
\end{aligned}
$$

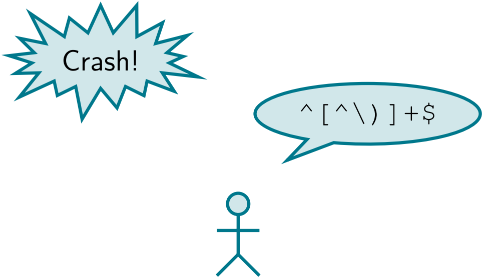
Expressiveness of Regular Expressions

Regular expressions describe regular languages:

- Every language described by a regular expression is a regular language.
  Proof: Structural induction on the syntax of regular expressions.
- Every regular language can be described using a regular expression.
  Proof: Standard translation of deterministic finite automata into regular expressions.

Limitations

Regular expressions sometimes look more like a swear word in a comic book than a specification of anything.



Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Specifying Correctness Properties

- Specifications must be easy to understand:
  Specification must be correct—otherwise verification makes no sense at all.
- We need kind of negation:
  It is often easier to specify the behaviour we do not want.

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Section

Linear Temporal Logic (LTL)

Propositional Logic

Temporal Logic

Chapter 2 “Specification Languages on Words”

Course “Runtime Verification”

M. Leucker & V. Stolz

INF5140 / V17

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING AND PROGRAMMING LANGUAGES

Another Idea

- A state of a system is a set of atomic propositions that hold in this state.
- An execution of a system is a finite sequence of such states.

Let's use operators of

- propositional logic to describe properties of one state.
- temporal logic to describe the relationship of states.
- propositional logic to combine this.

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

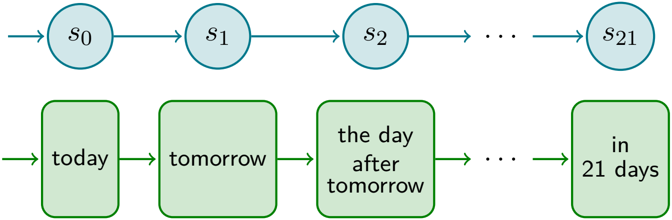
Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

A Simple Analogy

- A state is like a day.
- The initial state is like today.
- The next state is like tomorrow.



Remember

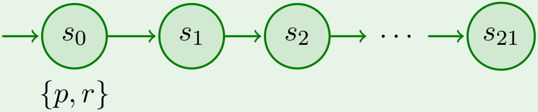
- A day is a state in the execution.
- A day is a letter in the word over $\Sigma = 2^{\mathrm{AP}}$.

Propositional Logic

Using propositional logic without temporal operators we describe only the first state (today).

Example

Consider $\mathrm{AP} = \{p, q, r, s\}$ and an initial state s_0 of an execution w in which p and r holds. We then have



$$w \models \text{true} \qquad w \not\models \text{false}$$

$$w \models p \qquad w \models p \wedge r \vee q$$

$$w \models \neg\, q \wedge \neg\, s \qquad w \not\models q.$$

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

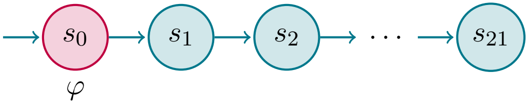
Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Formula: φ

The formula φ holds for an execution if φ holds in the first state s_0 of that execution.



Targets & Outline

Runs Are Words
States of the System
Executions Are Words

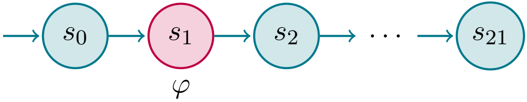
Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

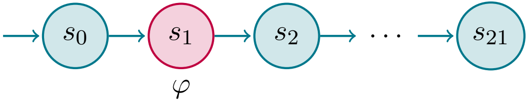
Next: $\mathrm{X}\,\varphi$

The formula $\mathrm{X}\,\varphi$ holds in state s_i if φ holds in state s_{i+1}.
If there is no state s_{i+1} then $\mathrm{X}\,\varphi$ never holds.



Weak Next: $\overline{\mathrm{X}}\,\varphi$

The formula $\overline{\mathrm{X}}\,\varphi$ holds in state s_i if φ holds in state s_{i+1}.
If there is no state s_{i+1} then $\overline{\mathrm{X}}\,\varphi$ always holds.



Targets & Outline

Runs Are Words
States of the System
Executions Are Words

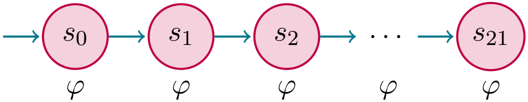
Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Globally: $\mathrm{G}\,\varphi$

The formula $\mathrm{G}\,\varphi$ holds in state s_i if φ holds in all states s_j for $j \geq i$.



Targets & Outline

Runs Are Words
- States of the System
- Executions Are Words

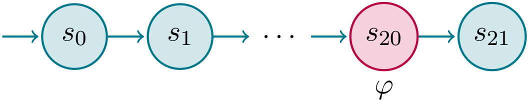
Regular Expressions
- The Idea
- Syntax and Semantics
- Limitations

Linear Temporal Logic (LTL)
- Propositional Logic
- Temporal Logic

Conclusion

Finally: $\mathrm{F}\,\varphi$

The formula $\mathrm{F}\,\varphi$ holds in state s_i if there is a state s_j for $j \geq i$ in which φ holds.



Targets & Outline

Runs Are Words
- States of the System
- Executions Are Words

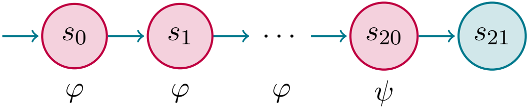
Regular Expressions
- The Idea
- Syntax and Semantics
- Limitations

Linear Temporal Logic (LTL)
- Propositional Logic
- Temporal Logic

Conclusion

Until: $\varphi \,\mathrm{U}\, \psi$

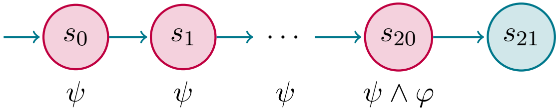
The formula $\varphi \,\mathrm{U}\, \psi$ holds in state s_i if there is a state s_j for $j \geq i$ in which ψ holds and φ holds in all states s_k for $i \leq k < j$.



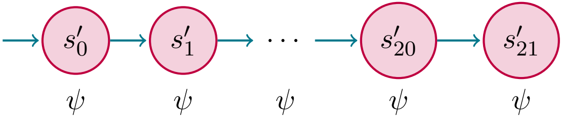
Notice that a state in which φ holds is not required in all cases!

Release: $\varphi \, \mathrm{R} \, \psi$

The formula $\varphi \, \mathrm{R} \, \psi$ holds in state s_i if there is a state s_j for $j \geq i$ in which φ holds and ψ holds in all states s_k for $i \leq k \leq j$.
If there is no such state s_j then the $\varphi \, \mathrm{R} \, \psi$ holds if ψ holds in all states s_k for $k \geq i$.



or



Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

Conclusion

1. The execution of a system is a word over the alphabet $\Sigma = 2^{\mathrm{AP}}$ where AP is the set of atomic propositions.
2. A correctness property is a language describing a set of executions.
3. Regular expressions describe regular languages and could be used to describe regular correctness properties.
4. Linear Temporal Logic (LTL) describes a subset of regular languages but is much better suited to describe correctness properties for runtime verification: Negation and Conjunction of LTL allows often to express correctness properties in a simple manner.

Targets & Outline

Runs Are Words
States of the System
Executions Are Words

Regular Expressions
The Idea
Syntax and Semantics
Limitations

Linear Temporal Logic (LTL)
Propositional Logic
Temporal Logic

Conclusion

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Chapter 3

LTL on Finite Words

Course "Runtime Verification"
M. Leucker & V. Stolz
INF5140 / V17

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING AND PROGRAMMING LANGUAGES

Chapter 3

Learning Targets of Chapter "LTL on Finite Words".

1. Learn about LTL.
2. Understand the LTL syntax.
3. Understand the LTL semantics on finite words: FLTL.
4. See how RV can be implemented using FLTL and learn about monitors for finite, terminated traces.

Chapter 3

Outline of Chapter “LTL on Finite Words”.

LTL Syntax
- LTL
- SALT

FLTL Semantics
- Semantics
- Examples and Equivalences
- Negation Normal Form

Monitor Function for FLTL
- The Idea
- Definition

Section

LTL Syntax

LTL
SALT

Chapter 3 “LTL on Finite Words”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

Recall: Specify Correctness Properties

Observing Executions

Targets & Outline

LTL Syntax
- LTL
- SALT

FLTL Semantics
- Semantics
- Examples and Equivalences
- Negation Normal Form

Monitor Function for FLTL
- The Idea
- Definition

Conclusion

Recall: Specify Correctness Properties

Observing Executions

Idea

Specify correctness properties in Linear Temporal Logic (LTL).

Targets & Outline

LTL Syntax
- LTL
- SALT

FLTL Semantics
- Semantics
- Examples and Equivalences
- Negation Normal Form

Monitor Function for FLTL
- The Idea
- Definition

Conclusion

Recall: Specify Correctness Properties

Observing Executions

Idea

Specify correctness properties in Linear Temporal Logic (LTL).

Commercial

Specify correctness properties in Regular Linear Temporal Logic (RLTL).

Targets & Outline

LTL Syntax
- LTL
- SALT

FLTL Semantics
- Semantics
- Examples and Equivalences
- Negation Normal Form

Monitor Function for FLTL
- The Idea
- Definition

Conclusion

Syntax of LTL Formulae

Definition (Syntax of LTL Formulae)

Let $p \in \mathrm{AP}$ be an atomic proposition from a finite set of atomic propositions AP. The set of LTL formulae is inductively defined by the following grammar:

$$\begin{array}{lll} \varphi & ::= & \mathrm{true} \mid p \mid \varphi \vee \varphi \mid \mathrm{X}\,\varphi \mid \varphi\,\mathrm{U}\,\varphi \mid \mathrm{F}\,\varphi \mid \\ & & \mathrm{false} \mid \neg p \mid \varphi \wedge \varphi \mid \overline{\mathrm{X}}\,\varphi \mid \varphi\,\mathrm{R}\,\varphi \mid \mathrm{G}\,\varphi \mid \\ & & \neg\,\varphi \end{array}$$

Order of Operations

The operator precedence is needed to determine an unambiguous derivation of an LTL formula if braces are left out in nested expressions. The higher the rank of an operator is the later it is derivated.

Braces only need to be added if an operator of lower or same rank should be derivated later than the current one.

Example (operator precedence of arithmetic)

1. exponential operator: $\bullet^{\bullet}$
2. multiplicative operators: $\cdot, /$
3. additive operators: $+, -$

Order of Operations

Definition (operator precedence of LTL)

1. negation operator: $\neg$
2. unary temporal operators: $\mathrm{X}, \overline{\mathrm{X}}, \mathrm{G}, \mathrm{F}$
3. binary temporal logic operators: U, R
4. conjunction operator: $\wedge$
5. disjunction operator: $\vee$

Example

$$\begin{aligned} & \mathrm{G}\ \neg x\ \vee\ \ \neg x\ \mathrm{U}\ \mathrm{G}\, y\ \ \wedge z \\ \equiv\ & \mathrm{G}\,(\neg x) \vee \Big((\,(\neg x)\,\mathrm{U}\,(\mathrm{G}\,y)\,) \wedge z \Big) \end{aligned}$$

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

LTL for the Working Engineer?

Simple?

LTL is for theoreticians—but for practitioners?

Targets & Outline

LTL Syntax

LTL

SALT

FLTL Semantics

Semantics

Examples and Equivalences

Negation Normal Form

Monitor Function for FLTL

The Idea

Definition

Conclusion

LTL for the Working Engineer?

Simple?

LTL is for theoreticians—but for practitioners?

SALT

Structured Assertion Language for Temporal Logic
$\Rightarrow$ Syntactic Sugar for LTL

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

www.isp.uni-luebeck.de/salt

Search

MY ACCOUNT IMPRESSUM

UNIVERSITY OF LÜBECK

INSTITUTE FOR SOFTWARE ENGINEERING AND PROGRAMMING LANGUAGES

isp

NEWS RESEARCH TEACHING STAFF CONTACT

Home > Research > Projects >

SALT - Smart Assertion Language for Temporal Logic

SALT

Goal

Do you want to specify the behavior of your program in a rigorously yet comfortable manner?
Do you see the benefits of temporal specifications but are bothered by the awkward formalisms available?
Do you want to use

- the power of a *Model Checker* to improve the quality of your systems or
- the powerful runtime reflection approach for bug hunting and elimination

but don't like the syntax of LTL?

If you can answer one of the above questions positively then **SALT is your solution!**

Try SALT click me

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Section

FLTL Semantics

Semantics
Examples and Equivalences
Negation Normal Form

Parts of Words

In the formal definition of LTL semantics we denote parts of a word as follows:

Let $w = a_1 a_2 \ldots a_n \in \Sigma^n$ be a finite word over the alphabet $\Sigma = 2^{\mathrm{AP}}$ and let $i \in \mathbb{N}$ with $1 \leq i \leq n$ be a position in this word. Then

- $|w| := n$ is the length of the word,
- $w_i = a_i$ is the i-th letter of the word and
- $w^i = a_i a_{i+1} \ldots a_n$ is the subword starting with letter i.

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

FLTL Semantics

Definition (FLTL Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$
\begin{array}{ll}
w \models \text{true} & \\
w \models p & \text{iff } p \in w_1 \\
w \models \neg p & \text{iff } p \notin w_1 \\
w \models \neg \varphi & \text{iff } w \not\models \varphi \\
w \models \varphi \vee \psi & \text{iff } w \models \varphi \text{ or } w \models \psi \\
w \models \varphi \wedge \psi & \text{iff } w \models \varphi \text{ and } w \models \psi
\end{array}
$$

FLTL Semantics

Definition (FLTL Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$
\begin{aligned}
w \models \mathrm{X}\,\varphi \quad & \text{iff } |w| > 1 \\
& \quad \text{and, for } |w| > 1, w^2 \models \varphi \\
w \models \overline{\mathrm{X}}\,\varphi \quad & \text{iff } |w| = 1 \\
& \quad \text{or, for } |w| > 1, w^2 \models \varphi
\end{aligned}
$$

Targets & Outline

LTL Syntax
- LTL
- SALT

FLTL Semantics
- Semantics
- Examples and Equivalences
- Negation Normal Form

Monitor Function for FLTL
- The Idea
- Definition

Conclusion

FLTL Semantics

Definition (FLTL Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$
\begin{aligned}
w \models \varphi \,\mathrm{U}\, \psi \quad & \text{iff } \exists i, 1 \leq i \leq |w| : (w^i \models \psi \\
& \quad \text{and } \forall k, 1 \leq k < i : w^k \models \varphi) \\
w \models \varphi \,\mathrm{R}\, \psi \quad & \text{iff } \exists i, 1 \leq i \leq |w| : (w^i \models \varphi \\
& \quad \text{and } \forall k, 1 \leq k \leq i : w^k \models \psi) \\
& \text{or } \forall i, 1 \leq i \leq |w| : w^i \models \psi
\end{aligned}
$$

Targets & Outline

LTL Syntax
- LTL
- SALT

FLTL Semantics
- Semantics
- Examples and Equivalences
- Negation Normal Form

Monitor Function for FLTL
- The Idea
- Definition

Conclusion

FLTL Semantics

Definition (FLTL Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$w \models \mathrm{F}\,\varphi \quad \text{iff } \exists i, 1 \leq i \leq |w| : w^i \models \varphi$$

$$w \models \mathrm{G}\,\varphi \quad \text{iff } \forall i, 1 \leq i \leq |w| : w^i \models \varphi$$

Targets & Outline

LTL Syntax
- LTL
- SALT

FLTL Semantics
- Semantics
- Examples and Equivalences
- Negation Normal Form

Monitor Function for FLTL
- The Idea
- Definition

Conclusion

Finally and Globally Examples

Examples (Finally and Globally)

Consider words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset \models \mathrm{F}\, q$.

Finally and Globally Examples

Examples (Finally and Globally)

Consider words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset \models \mathrm{F}\, q$.
- $\{q\}\{q\}\{p, q\}\{q\}\{q\} \models \mathrm{G}\, q$.

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

Finally and Globally Examples

Examples (Finally and Globally)

Consider words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset \models \mathrm{F}\, q$.
- $\{q\}\{q\}\{p,q\}\{q\}\{q\} \models \mathrm{G}\, q$.
- $\emptyset\{p\}\{p,q\}\emptyset\{q\}\emptyset\{q\} \models \mathrm{G}\,\mathrm{F}\, q$.

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

Finally and Globally Examples

Examples (Finally and Globally)

Consider words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset \models \mathrm{F}\, q$.
- $\{q\}\{q\}\{p,q\}\{q\}\{q\} \models \mathrm{G}\, q$.
- $\emptyset\{p\}\{p,q\}\emptyset\{q\}\emptyset\{q\} \models \mathrm{G}\,\mathrm{F}\, q$.
- $\{p\}\emptyset\{q\}\{p\}\{p,q\}\{p,q\}\{q\} \models \mathrm{F}\,\mathrm{G}\, q$.

Finally and Globally Examples

Examples (Finally and Globally)

Consider words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset \models \mathrm{F}\, q$.
- $\{q\}\{q\}\{p,q\}\{q\}\{q\} \models \mathrm{G}\, q$.
- $\emptyset\{p\}\{p,q\}\emptyset\{q\}\emptyset\{q\} \models \mathrm{G}\,\mathrm{F}\, q$.
- $\{p\}\emptyset\{q\}\{p\}\{p,q\}\{p,q\}\{q\} \models \mathrm{F}\,\mathrm{G}\, q$.

- $\mathrm{G}\,\mathrm{F}\,\varphi$ can be read as: For every state (globally) there will be a state in the future (finally) in that φ holds.
- $\mathrm{F}\,\mathrm{G}\,\varphi$ can be read as: There will be a state in the future (finally) that φ holds in every state (globally).

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

Practical Examples

In the following examples we consider these scopes:

everytime: all states

before ψ: all states before the first state in which ψ holds (if there is such a state)

after ψ: all states after and including the first state in which ψ holds (if there is such a state)

Example (Absence)

The formula φ does not hold

everytime: $\mathrm{G}\,\neg\,\varphi$

before ψ: $(\mathrm{F}\,\psi) \rightarrow (\neg\,\varphi\,\mathrm{U}\,\psi)$

after ψ: $\mathrm{G}(\psi \rightarrow (\mathrm{G}\,\neg\,\varphi))$

Practical Examples

In the following examples we consider these scopes:

everytime: all states

before ψ: all states before the first state in which ψ holds (if there is such a state)

after ψ: all states after and including the first state in which ψ holds
(if there is such a state)

Example (Existence)

The formula φ holds in the future

everytime: $\mathrm{F}\,\varphi$

before ψ: $\mathrm{G}\,\neg\,\psi \vee \neg\,\psi\,\mathrm{U}(\varphi \wedge \neg\,\psi)$

after ψ: $\mathrm{G}\,\neg\,\psi \vee \mathrm{F}(\psi \wedge \mathrm{F}\,\varphi)$

Targets & Outline
LTL Syntax
LTL
SALT
FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form
Monitor Function for FLTL
The Idea
Definition
Conclusion

Practical Examples

In the following examples we consider these scopes:

everytime: all states

before ψ: all states before the first state in which ψ holds (if there is such a state)

after ψ: all states after and including the first state in which ψ holds (if there is such a state)

Example (Universality)

The formula φ holds

everytime: $\mathrm{G}\,\varphi$

before ψ: $(\mathrm{F}\,\psi) \rightarrow (\varphi\,\mathrm{U}\,\psi)$

after ψ: $\mathrm{G}(\psi \rightarrow \mathrm{G}\,\varphi)$

Equivalences

Definition (Equivalence of Formulae)

Let $\Sigma = 2^{\mathrm{AP}}$ and φ and ψ be LTL formulae over AP. φ and ψ are *equivalent*, denoted by $\varphi \equiv \psi$, iff

$$\forall w \in \Sigma^{+} : w \models \varphi \Leftrightarrow w \models \psi.$$

Globally and finally can easily be expressed using until and release:

$$\mathrm{F}\,\varphi \equiv \mathrm{true}\,\mathrm{U}\,\varphi$$
$$\mathrm{G}\,\varphi \equiv \mathrm{false}\,\mathrm{R}\,\varphi$$

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

De Morgan Rules

The negation can always be moved in front of the atomic propositions using the dual operators:

De Morgan Rules of Propositional Logic

$$\neg(\varphi \vee \psi) \equiv \neg\,\varphi \wedge \neg\,\psi$$
$$\neg(\varphi \wedge \psi) \equiv \neg\,\varphi \vee \neg\,\psi$$

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

De Morgan Rules

The negation can always be moved in front of the atomic propositions using the dual operators:

De Morgan Rules of Temporal Logic

$$\neg(\varphi \, \mathrm{U} \, \psi) \equiv \neg\,\varphi \, \mathrm{R} \, \neg\,\psi$$

$$\neg(\varphi \, \mathrm{R} \, \psi) \equiv \neg\,\varphi \, \mathrm{U} \, \neg\,\psi$$

$$\neg(\mathrm{G}\,\varphi) \equiv \mathrm{F}\,\neg\,\varphi$$

$$\neg(\mathrm{F}\,\varphi) \equiv \mathrm{G}\,\neg\,\varphi$$

$$\neg(\mathrm{X}\,\varphi) \equiv \overline{\mathrm{X}}\,\neg\,\varphi$$

$$\neg(\overline{\mathrm{X}}\,\varphi) \equiv \mathrm{X}\,\neg\,\varphi$$

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

Fixed Point Equations

The following fixed point equations can be used to step-wise unwind until and release:

$$\varphi \, \mathrm{U} \, \psi \equiv \psi \vee (\varphi \wedge \mathrm{X}(\varphi \, \mathrm{U} \, \psi))$$
$$\varphi \, \mathrm{R} \, \psi \equiv \psi \wedge (\varphi \vee \overline{\mathrm{X}}(\varphi \, \mathrm{R} \, \psi))$$

Consequently such fix point equations for globally and finally are special cases of the above ones:

$$\mathrm{G} \, \varphi \equiv \varphi \wedge \overline{\mathrm{X}}(\mathrm{G} \, \varphi)$$
$$\mathrm{F} \, \varphi \equiv \varphi \vee \mathrm{X}(\mathrm{F} \, \varphi)$$

Negation Normal Form (NNF)

Definition (Negation Normal Form (NNF))

An LTL formula φ is in Negation Normal Form (NNF) iff $\neg$ only occurs in front of atomic propositions $p \in \mathrm{AP}$.

Targets & Outline

LTL Syntax
- LTL
- SALT

FLTL Semantics
- Semantics
- Examples and Equivalences
- Negation Normal Form

Monitor Function for FLTL
- The Idea
- Definition

Conclusion

Negation Normal Form (NNF)

Definition (Negation Normal Form (NNF))

An LTL formula φ is in Negation Normal Form (NNF) iff $\neg$ only occurs in front of atomic propositions $p \in \mathrm{AP}$.

Lemma

For every LTL formula there exists an equivalent formula in NNF.

Proof.

Recursively apply De Morgan rules of propositional logic and De Morgan rules of temporal logic. □

Section

Monitor Function for FLTL

The Idea

Definition

Chapter 3 “LTL on Finite Words”

Course “Runtime Verification”

M. Leucker & V. Stolz

INF5140 / V17

The Idea

Build up a function that

- takes an LTL formula φ in NNF and a word $w \in \Sigma^+$,
- performs recursion on the structure of φ
- returns true iff $w \models \varphi$.

First Ideas

Let $p \in \mathrm{AP}$ be an atomic proposition and $w \in \Sigma^+$ a word.

We then can evaluate

- true and false.
- $\varphi \vee \psi$ by evaluating φ, evaluating ψ and computing $\varphi \vee \psi$.
- p by checking if $p \in w_1$.
- $\neg p$ by checking if $p \notin w_1$.

Further Ideas

What about next?

We can check if $w \models \mathrm{X}\,\varphi$ holds by omitting

- the first letter of w and
- the next operator

and checking if $w^2 \models \varphi$ holds.

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

Further Ideas

What about next?

We can check if $w \models \mathrm{X}\,\varphi$ holds by omitting

- the first letter of w and
- the next operator

and checking if $w^2 \models \varphi$ holds.

What about until and release?

Use the already presented fixpoint equations and the above ideas to evaluate conjunction, disjunction and next.

$$\varphi\,\mathrm{U}\,\psi \equiv \psi \vee (\varphi \wedge \mathrm{X}(\varphi\,\mathrm{U}\,\psi))$$
$$\varphi\,\mathrm{R}\,\psi \equiv \psi \wedge (\varphi \vee \overline{\mathrm{X}}(\varphi\,\mathrm{R}\,\psi))$$

evlFLTL

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $w \in \Sigma^+$ a finite non-empty word, φ and ψ LTL formulae and $\mathbb{B}_2 = \{\top, \bot\}$.

We then define the function $\mathrm{evlFLTL} : \Sigma^+ \times \mathrm{LTL} \to \mathbb{B}_2$ inductively as follows:

$$
\begin{aligned}
\mathrm{evlFLTL}(w, \mathrm{true}) &= \top \\
\mathrm{evlFLTL}(w, \mathrm{false}) &= \bot \\
\mathrm{evlFLTL}(w, \varphi \vee \psi) &= \mathrm{evlFLTL}(w, \varphi) \vee \mathrm{evlFLTL}(w, \psi) \\
\mathrm{evlFLTL}(w, \varphi \wedge \psi) &= \mathrm{evlFLTL}(w, \varphi) \wedge \mathrm{evlFLTL}(w, \psi) \\
\mathrm{evlFLTL}(w, p) &= (p \in w_1) \\
\mathrm{evlFLTL}(w, \neg p) &= (p \notin w_1)
\end{aligned}
$$

evlFLTL

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $w \in \Sigma^+$ a finite non-empty word, φ and ψ LTL formulae and $\mathbb{B}_2 = \{\top, \bot\}$.

We then define the function $\mathrm{evlFLTL} : \Sigma^+ \times \mathrm{LTL} \to \mathbb{B}_2$ inductively as follows:

$$\begin{aligned}
\mathrm{evlFLTL}(w, \varphi \,\mathrm{U}\, \psi) &= \mathrm{evlFLTL}(w, \psi \vee (\varphi \wedge \mathrm{X}(\varphi \,\mathrm{U}\, \psi))) \\
\mathrm{evlFLTL}(w, \varphi \,\mathrm{R}\, \psi) &= \mathrm{evlFLTL}(w, \psi \wedge (\varphi \vee \overline{\mathrm{X}}(\varphi \,\mathrm{R}\, \psi))) \\
\mathrm{evlFLTL}(w, \mathrm{F}\, \varphi) &= \mathrm{evlFLTL}(w, \varphi \vee \mathrm{X}\,\mathrm{F}\, \varphi) \\
\mathrm{evlFLTL}(w, \mathrm{G}\, \varphi) &= \mathrm{evlFLTL}(w, \varphi \wedge \overline{\mathrm{X}}\,\mathrm{G}\, \varphi) \\
\mathrm{evlFLTL}(w, \mathrm{X}\, \varphi) &= (|w| > 1) \wedge \mathrm{evlFLTL}(w^2, \varphi) \\
\mathrm{evlFLTL}(w, \overline{\mathrm{X}}\, \varphi) &= (|w| = 1) \vee \mathrm{evlFLTL}(w^2, \varphi)
\end{aligned}$$

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

Conclusion

1. The LTL operations negation, disjunction, until and next are enough to gain the full expressiveness of LTL.
2. If you add the dual operators every LTL formula has a negation normal form (NNF).
3. The fix point equations can be used to step-wise unwind until and release using next and weak next.
4. $\mathrm{evlFLTL}$ is as inductively defined function that answers the question if a given finite non-empty word models a correctness property given as an LTL formula in NNF and can easily be implemented recursively.

Targets & Outline

LTL Syntax
LTL
SALT

FLTL Semantics
Semantics
Examples and Equivalences
Negation Normal Form

Monitor Function for FLTL
The Idea
Definition

Conclusion

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Chapter 4

Impartial Runtime Verification

Course "Runtime Verification"

M. Leucker & V. Stolz

INF5140 / V17

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING AND PROGRAMMING LANGUAGES

Chapter 4

Learning Targets of Chapter "Impartial Runtime Verification".

1. Understand the idea of impartiality and why we want to use impartial evaluation of LTL formulae.
2. Learn the basics of truth domains and lattices.
3. Understand the four-valued LTL semantics on finite words: FLTL_4.
4. See how impartial RV can be implemented using FLTL_4 and learn about automata based monitors for finite, non-completed traces.

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Chapter 4

Outline of Chapter “Impartial Runtime Verification”.

Truth Domains

Motivation

Definition

Four-Valued LTL Semantics: FLTL_4

Definition

Monitor Function

Mealy Machines

Determinstic Mealy Machines

Alternating Mealy Machines

Automata Based RV

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Section

Truth Domains

Motivation
Definition

Chapter 4 “Impartial Runtime Verification”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

Words Aren't Terminated—They Are Growing

- In the last chapters we considered finite terminated words.
- A monitor for RV does not get a formula and a finite terminated word.
- A monitor for RV gets a formula and one letter after another.
- With every new system state the monitor gets one more letter.

Examples of Evaluating Growing Words

Consider

- the alphabet $\Sigma = 2^{\mathrm{AP}}$ where $\mathrm{AP} = \{p, q\}$,
- the properties $\varphi = \mathrm{G}\, p$ and $\varphi' = \mathrm{F}\, p$ and
- words w and w' growing with every new state.

Lets watch monitors for RV at work:

$$w = \{p, q\}$$

$$w \models \varphi$$

Examples of Evaluating Growing Words

Consider

- the alphabet $\Sigma = 2^{\mathrm{AP}}$ where $\mathrm{AP} = \{p, q\}$,
- the properties $\varphi = \mathrm{G}\, p$ and $\varphi' = \mathrm{F}\, p$ and
- words w and w' growing with every new state.

Lets watch monitors for RV at work:

$$w = \{p, q\}\ \{p\}$$

$w \models \varphi$

$w \models \varphi$

Examples of Evaluating Growing Words

Consider

- the alphabet $\Sigma = 2^{\mathrm{AP}}$ where $\mathrm{AP} = \{p, q\}$,
- the properties $\varphi = \mathrm{G}\, p$ and $\varphi' = \mathrm{F}\, p$ and
- words w and w' growing with every new state.

Lets watch monitors for RV at work:

$$w = \{p,q\}\ \{p\}\ \{q\}$$

$w \models \varphi$

$w \models \varphi$

$w \not\models \varphi$

Examples of Evaluating Growing Words

Consider

- the alphabet $\Sigma = 2^{\mathrm{AP}}$ where $\mathrm{AP} = \{p, q\}$,
- the properties $\varphi = \mathrm{G}\, p$ and $\varphi' = \mathrm{F}\, p$ and
- words w and w' growing with every new state.

Lets watch monitors for RV at work:

$$w = \{p, q\}\ \{p\}\ \{q\}\ \{p\}$$

$w \models \varphi$

$w \models \varphi$

$w \not\models \varphi$

$w \not\models \varphi$

Targets & Outline

Truth Domains
Motivation
Definition

Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function

Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Conclusion

Examples of Evaluating Growing Words

Consider

- the alphabet $\Sigma = 2^{\mathrm{AP}}$ where $\mathrm{AP} = \{p, q\}$,
- the properties $\varphi = \mathrm{G}\, p$ and $\varphi' = \mathrm{F}\, p$ and
- words w and w' growing with every new state.

Lets watch monitors for RV at work:

$w = \{p, q\}\ \{p\}\ \{q\}\ \{p\}$ $\qquad$ $w' = \{q\}$

$w \models \varphi$ $\qquad$ $w' \not\models \varphi'$

$w \models \varphi$

$w \not\models \varphi$

$w \not\models \varphi$

Targets & Outline

Truth Domains

Motivation

Definition

Four-Valued LTL Semantics: FLTL_4

Definition

Monitor Function

Mealy Machines

Determinstic Mealy Machines

Alternating Mealy Machines

Automata Based RV

Conclusion

Examples of Evaluating Growing Words

Consider

- the alphabet $\Sigma = 2^{\mathrm{AP}}$ where $\mathrm{AP} = \{p, q\}$,
- the properties $\varphi = \mathrm{G}\, p$ and $\varphi' = \mathrm{F}\, p$ and
- words w and w' growing with every new state.

Lets watch monitors for RV at work:

$w = \{p, q\}\ \{p\}\ \{q\}\ \{p\}$

$w \models \varphi$

$w \models \varphi$

$w \not\models \varphi$

$w \not\models \varphi$

$w' = \{q\}\ \{q\}$

$w' \not\models \varphi'$

$w' \not\models \varphi'$

Examples of Evaluating Growing Words

Consider

- the alphabet $\Sigma = 2^{\mathrm{AP}}$ where $\mathrm{AP} = \{p, q\}$,
- the properties $\varphi = \mathrm{G}\, p$ and $\varphi' = \mathrm{F}\, p$ and
- words w and w' growing with every new state.

Lets watch monitors for RV at work:

$w = \{p, q\}\ \{p\}\ \{q\}\ \{p\}$

$w \models \varphi$

$w \models \varphi$

$w \not\models \varphi$

$w \not\models \varphi$

$w' = \{q\}\ \{q\}\ \{p, q\}$

$w' \not\models \varphi'$

$w' \not\models \varphi'$

$w' \models \varphi'$

Targets & Outline
Truth Domains
Motivation
Definition
Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function
Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV
Conclusion

Examples of Evaluating Growing Words

Consider

- the alphabet $\Sigma = 2^{\mathrm{AP}}$ where $\mathrm{AP} = \{p, q\}$,
- the properties $\varphi = \mathrm{G}\, p$ and $\varphi' = \mathrm{F}\, p$ and
- words w and w' growing with every new state.

Lets watch monitors for RV at work:

$$w = \{p,q\}\ \{p\}\ \{q\}\ \{p\} \qquad w' = \{q\}\ \{q\}\ \{p,q\}\ \{p\}$$

w	w'
$\{p,q\}$: $w \models \varphi$	$\{q\}$: $w' \not\models \varphi'$
$\{p,q\}\ \{p\}$: $w \models \varphi$	$\{q\}\ \{q\}$: $w' \not\models \varphi'$
$\{p,q\}\ \{p\}\ \{q\}$: $w \not\models \varphi$	$\{q\}\ \{q\}\ \{p,q\}$: $w' \models \varphi'$
$\{p,q\}\ \{p\}\ \{q\}\ \{p\}$: $w \not\models \varphi$	$\{q\}\ \{q\}\ \{p,q\}\ \{p\}$: $w' \models \varphi'$

Targets & Outline
Truth Domains
Motivation
Definition
Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function
Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV
Conclusion

Impartiality

Be Impartial!

- go for a final verdict ($\top$ or $\bot$) only if you really know
- be a rational being: stick to your word

Definition (Impartiality)

Impartiality requires that a finite trace is not evaluated to $true$ or, respectively $false$, if there still exists an (possibly infinite) continuation leading to another verdict.

Semantic Function

Definition (Semantic Function)

The semantic function

$$\mathrm{sem}_k : \Sigma^+ \times \mathrm{LTL} \to \mathbb{B}_k$$

maps a word $w \in \Sigma^+$ and a an LTL formula φ to a logic value $b \in \mathbb{B}_k$.

We use $[\![w \models \varphi]\!]_k = b$ instead of $\mathrm{sem}_k(w, \varphi) = b$.

Semantic Function for FLTL

- We defined the FLTL semantics as relation $w \models \varphi$ between a word $w \in \Sigma^+$ and an LTL formula φ.
- This can be interpreted as semantic function

$$\mathrm{sem}_2 : \Sigma^+ \times \mathrm{LTL} \to \mathbb{B}_2,$$

$$\mathrm{sem}_2(w, \varphi) = [\![w \models \varphi]\!]_2 := \begin{cases} \top & \text{if } w \models \varphi \\ \bot & \text{else.} \end{cases}$$

Targets & Outline

Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

Impartiality

Definition (Impartial Semantics)

Let $\Sigma = 2^{\mathrm{AP}}$ be an alphabet, $w \in \Sigma^+$ a word and φ an LTL formula. A semantic function is called impartial iff for all $u \in \Sigma^*$

$$[\![w \models \varphi]\!] = \top \text{ implies } [\![wu \models \varphi]\!] = \top$$
$$[\![w \models \varphi]\!] = \bot \text{ implies } [\![wu \models \varphi]\!] = \bot.$$

Target

Create monitors which only answer $\top$ or $\bot$ if the result keeps stable for a growing word.

We Need Multiple Values

FLTL semantics are not impartial:

$$w = \{a\}, \varphi = \mathrm{G}\, a \text{ and } wu = \{a\}\{b\}$$

is a counterexample for

$$[\![w \models \varphi]\!] = \top \text{ implies } [\![wu \models \varphi]\!] = \top.$$

Targets & Outline

Truth Domains

Motivation

Definition

Four-Valued LTL Semantics: FLTL_4

Definition

Monitor Function

Mealy Machines

Determinstic Mealy Machines

Alternating Mealy Machines

Automata Based RV

Conclusion

We Need Multiple Values

FLTL semantics are not impartial:

$$w = \{a\}, \varphi = \mathrm{G}\, a \text{ and } wu = \{a\}\{b\}$$

is a counterexample for

$$[\![w \models \varphi]\!] = \top \text{ implies } [\![wu \models \varphi]\!] = \top.$$

Impartiality implies multiple values

Every two-valued logic is not impartial.

- Impartiality forbids switching from $\top$ to $\bot$ and vice versa.
- Therefore we need more logic values than $\top$ and $\bot$.

Lattice

Definition (Lattice)

A *lattice* is a partially ordered set $(\mathcal{L}, \sqsubseteq)$ where for each $x, y \in \mathcal{L}$, there exists

1. a unique *greatest lower bound* (glb), which is called the *meet* of x and y, and is denoted with $x \sqcap y$, and
2. a unique *least upper bound* (lub), which is called the *join* of x and y, and is denoted with $x \sqcup y$.

If the ordering relation $\sqsubseteq$ is obvious we denote the lattice with the set $\mathcal{L}$.

Targets & Outline
Truth Domains
Motivation
Definition
Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function
Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV
Conclusion

Finite Lattice

Definition (Finite Lattice)

A lattice $(\mathcal{L}, \sqsubseteq)$ is called *finite* iff $\mathcal{L}$ is finite.

Every non-empty finite lattice has two well-defined unique elements:

- A least element, called *bottom*, denoted with $\perp$ and
- a greatest element, called *top*, denoted with $\top$.

Targets & Outline

Truth Domains
Motivation
Definition

Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function

Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Conclusion

Hasse diagram

- Hasse diagrams are used to represent a finite partially ordered set.
- Each element of the set is represented as a vertex in the plane.
- For all $x, y \in \mathcal{L}$ where $x \sqsubseteq y$ but no $z \in \mathcal{L}$ exists where $x \sqsubseteq z \sqsubseteq y$ a line that goes upward from x to y is drawn.

Example

Hasse diagram for $\mathbb{B}_2 = \{\bot, \top\}$ with $\bot \sqsubseteq \top$:

$$\begin{array}{c} \top \\ | \\ \bot \end{array}$$

Targets & Outline

Truth Domains

Motivation

Definition

Four-Valued LTL Semantics: FLTL_4

Definition

Monitor Function

Mealy Machines

Determinstic Mealy Machines

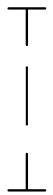
Alternating Mealy Machines

Automata Based RV

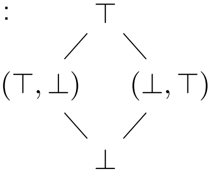
Conclusion

Example Lattices

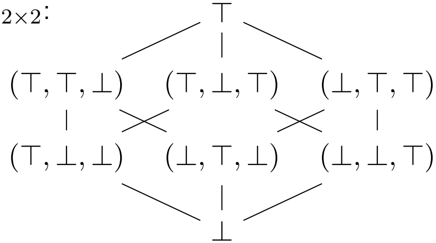
$\mathbb{B}_2$:



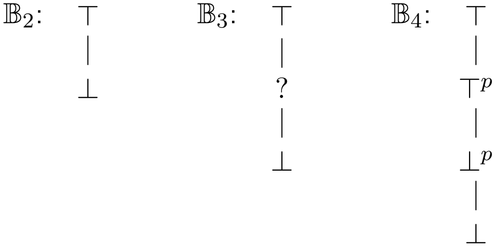
$\mathbb{B}_{2\times 2}$:



$\mathbb{B}_{2\times 2\times 2}$:



Example Lattices II



Distributive Lattices

Definition (Distributive Lattices)

A lattice $(\mathcal{L}, \sqsubseteq)$ is called a *distributive lattice* iff we have for all elements $x, y, z \in \mathcal{L}$

$$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z) \text{ and}$$
$$x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z).$$

Targets & Outline

Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

De Morgan Lattice

Definition (De Morgan Lattice)

A distributive lattice $(\mathcal{L}, \sqsubseteq)$ is called a *De Morgan lattice* iff every element $x \in \mathcal{L}$ has a unique *dual* element $\overline{x}$, such that

$$\overline{\overline{x}} = x \text{ and } x \sqsubseteq y \text{ implies } \overline{y} \sqsubseteq \overline{x}.$$

Boolean Lattice

Definition (Boolean Lattice)

A De Morgan lattice is called *Boolean lattice* iff for every element x and its dual element $\overline{x}$ we have

$$x \sqcup \overline{x} = \top \text{ and } x \sqcap \overline{x} = \bot.$$

Every Boolean lattice has 2^n elements for some $n \in \mathbb{N}$.

Truth Domain

Definition (Truth Domain)

A *Truth Domain* is a finite De Morgan Lattice.

Examples (Truth Domains)

The following lattices are all Truth Domains:

- $\mathbb{B}_2 = \{\top, \bot\}$ with $\bot \sqsubseteq \top$ and $\overline{\top} = \bot$ and $\overline{\bot} = \top$.
- $\mathbb{B}_3 = \{\top, ?, \bot\}$ with $\bot \sqsubseteq ? \sqsubseteq \top$ and $\overline{\top} = \bot$, $\overline{?} = ?$ and $\overline{\bot} = \top$.
- $\mathbb{B}_4 = \{\top, \top^p, \bot^p, \bot\}$ with $\bot \sqsubseteq \bot^p \sqsubseteq \top^p \sqsubseteq \top$ and $\overline{\top} = \bot$, $\overline{\top^p} = \bot^p$, $\overline{\bot^p} = \top^p$ and $\overline{\bot} = \top$.

Targets & Outline

Truth Domains

Motivation

Definition

Four-Valued LTL Semantics: FLTL_4

Definition

Monitor Function

Mealy Machines

Determinstic Mealy Machines

Alternating Mealy Machines

Automata Based RV

Conclusion

Section

Four-Valued LTL Semantics: FLTL_4

Definition

Monitor Function

Chapter 4 "Impartial Runtime Verification"
Course "Runtime Verification"
M. Leucker & V. Stolz
INF5140 / V17

Examples of Impartial LTL Semantics

- We want to create impartial four-valued semantics for LTL on finite, non-completed words
- using the truth domain $(\mathbb{B}_4, \sqsubseteq)$.

Examples (FLTL vs. FLTL$_4$)

The indices 2 and 4 denote FLTL resp. FLTL$_4$.

$$
\begin{aligned}
[\![\emptyset \models \mathrm{X}\, a]\!]_2 &= \bot & \qquad [\![\emptyset \models \mathrm{X}\, a]\!]_4 &= \bot^p \\
[\![\emptyset\emptyset \models \mathrm{X}\, a]\!]_2 &= \bot & \qquad [\![\emptyset\emptyset \models \mathrm{X}\, a]\!]_4 &= \bot \\
[\![\emptyset\{a\} \models \mathrm{X}\, a]\!]_2 &= \top & \qquad [\![\emptyset\{a\} \models \mathrm{X}\, a]\!]_4 &= \top \\
[\![\emptyset \models \overline{\mathrm{X}}\, a]\!]_2 &= \top & \qquad [\![\emptyset \models \overline{\mathrm{X}}\, a]\!]_4 &= \top^p \\
[\![\emptyset\emptyset \models \overline{\mathrm{X}}\, a]\!]_2 &= \bot & \qquad [\![\emptyset\emptyset \models \overline{\mathrm{X}}\, a]\!]_4 &= \bot \\
[\![\emptyset\{a\} \models \overline{\mathrm{X}}\, a]\!]_2 &= \top & \qquad [\![\emptyset\{a\} \models \overline{\mathrm{X}}\, a]\!]_4 &= \top
\end{aligned}
$$

How To Create Impartial LTL Semantics?

At the end of the word

- X evaluates to $\bot^p$ instead of $\bot$ and
- $\overline{\mathrm{X}}$ evaluates to $\top^p$ instead of $\top$.

Idea of $\bullet^p$: Semantics if the word ends here.

Fulfilling the introduced equivalences and fix point equations we get at the end of the word:

- U evaluates to $\bot^p$ instead of $\bot$,
- R evaluates to $\top^p$ instead of $\top$,
- F evaluates to $\bot^p$ instead of $\bot$ and
- G evaluates to $\top^p$ instead of $\top$.

Idea of $\bullet^p$: Semantics if the word ends here or goes on like this forever.

Properties of $(\mathbb{B}_4, \sqsubseteq)$

- $(\mathbb{B}_4, \sqsubseteq)$ is no Boolean Lattice.
- Some equivalences in FLTL do not hold in FLTL_4.

For any LTL formula φ using FLTL semantics we have

$$\varphi \vee \neg\varphi \equiv_2 \text{true} \text{ and } \varphi \wedge \neg\varphi \equiv_2 \text{false}.$$

Examples (FLTL vs. FLTL$_4$)

For any $w \in \Sigma^+$ and $a \in \Sigma$ we have

$$[\![w \models \mathrm{G}\, a \vee \neg\, \mathrm{G}\, a]\!]_2 = \top \qquad [\![w \models \mathrm{G}\, a \vee \neg\, \mathrm{G}\, a]\!]_4 = \top^p$$

$$[\![w \models \mathrm{F}\, a \wedge \neg\, \mathrm{F}\, a]\!]_2 = \bot \qquad [\![w \models \mathrm{F}\, a \wedge \neg\, \mathrm{F}\, a]\!]_4 = \bot^p$$

FLTL$_4$ Semantics

Definition (FLTL$_4$ Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$[\![w \models \text{true}]\!]_4 = \top$$

$$[\![w \models \text{false}]\!]_4 = \bot$$

$$[\![w \models p]\!]_4 = \begin{cases} \top & \text{if } p \in w_1 \\ \bot & \text{if } p \notin w_1 \end{cases}$$

$$[\![w \models \neg p]\!]_4 = \begin{cases} \top & \text{if } p \notin w_1 \\ \bot & \text{if } p \in w_1 \end{cases}$$

FLTL$_4$ Semantics

Definition (FLTL$_4$ Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$\begin{aligned}
[\![w \models \neg \varphi]\!]_4 &= \overline{[\![w \models \varphi]\!]_4} \\
[\![w \models \varphi \vee \psi]\!]_4 &= [\![w \models \varphi]\!]_4 \sqcup [\![w \models \psi]\!]_4 \\
[\![w \models \varphi \wedge \psi]\!]_4 &= [\![w \models \varphi]\!]_4 \sqcap [\![w \models \psi]\!]_4
\end{aligned}$$

FLTL$_4$ Semantics

Definition (FLTL$_4$ Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$[\![w \models \mathrm{X}\,\varphi]\!]_4 = \begin{cases} [\![w^2 \models \varphi]\!]_4 & \text{if } |w| > 1 \\ \bot^p & \text{else} \end{cases}$$

$$[\![w \models \overline{\mathrm{X}}\,\varphi]\!]_4 = \begin{cases} [\![w^2 \models \varphi]\!]_4 & \text{if } |w| > 1 \\ \top^p & \text{else} \end{cases}$$

FLTL_4 Semantics

Definition (FLTL_4 Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$
\begin{aligned}
&[\![w \models \varphi \,\mathrm{U}\, \psi]\!]_4 \\
&\quad = \left(\bigsqcup_{1 \leq i \leq |w|} \left([\![w^i \models \psi]\!]_4 \sqcap \bigsqcap_{1 \leq j < i} [\![w^j \models \varphi]\!]_4 \right) \right) \\
&\qquad \sqcup \left(\bot^p \sqcap \bigsqcap_{1 \leq i \leq |w|} [\![w^i \models \varphi]\!]_4 \right)
\end{aligned}
$$

Targets & Outline

Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

FLTL_4 Semantics

Definition (FLTL_4 Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$\begin{aligned}
&[\![w \models \varphi \,\mathrm{R}\, \psi]\!]_4 \\
&\quad = \left(\bigsqcup_{1 \leq i \leq |w|} \left([\![w^i \models \varphi]\!]_4 \sqcap \bigsqcap_{1 \leq j \leq i} [\![w^j \models \psi]\!]_4 \right) \right) \\
&\qquad\qquad \sqcup \left(\top^p \sqcap \bigsqcap_{1 \leq i \leq |w|} [\![w^i \models \psi]\!]_4 \right)
\end{aligned}$$

FLTL_4 Semantics

Definition (FLTL_4 Semantics)

Let φ, ψ be LTL formulae and let $w \in \Sigma^+$ be a finite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$[\![w \models \mathrm{F}\,\varphi]\!]_4 = \bot^p \sqcup \bigsqcup_{1 \leq i \leq |w|} [\![w^i \models \varphi]\!]_4$$

$$[\![w \models \mathrm{G}\,\varphi]\!]_4 = \top^p \sqcap \bigsqcap_{1 \leq i \leq |w|} [\![w^i \models \varphi]\!]_4$$

Targets & Outline

Truth Domains
Motivation
Definition

Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function

Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Conclusion

Equivalences

Definition (Equivalence of Formulae)

Let $\Sigma = 2^{\mathrm{AP}}$ and φ and ψ be LTL formulae over AP. φ and ψ are *equivalent*, denoted by $\varphi \equiv \psi$, iff

$$\forall w \in \Sigma^+ : [\![w \models \varphi]\!] = [\![w \models \psi]\!].$$

The equivalences desribed in the previous chapter are still valid using the semantic function of FLTL_4.

Monitor Function

Left-to-right!

Targets & Outline

Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

The Idea

Build up a monitor function for evaluating each subsequent letter of non-completed words.

Such a function

- takes an LTL formula φ in NNF and a letter $a \in \Sigma$,
- performs (not recursive) formula rewriting (progression) and
- returns $[\![a \models \varphi]\!]_4$ and a new LTL formula φ' that the next letter has to fulfill.

The Idea of Progression

- Compute only the semantics of the first letter and let someone else do the rest.
- Rewrite the LTL formula to keep track of what is done and what still needs to be checked.
- Thanks to the impartial semantics we don't need to know the whole word to compute a valid semantics.

Examples

Let $w \in \Sigma^+$ be a word and $a \in \Sigma$ a letter.

- We can compute $[\![w \models \mathrm{X}\, a]\!]_4$ by doing nothing and letting someone else check $[\![w^2 \models a]\!]_4$.
- We can compute $[\![w \models a]\!]_4$ by checking $a \in w_1$. Then the LTL formula is over. This is denoted by true or false as new formula.

Further Ideas

We know how to evaluate

- atomic propositions,
- positive operators of propositional logic ($\wedge, \vee$) and
- next-formulas.

That's it thanks to

- equivalences for G and F,
- De Morgan rules of propositional and temporal logic for negation ($\neg$) and
- and fixed point equations for U and R.

$\mathrm{evlFLTL}_4$

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $a \in \Sigma$ a letter, and φ and ψ LTL formulae.

We then define the function
$\mathrm{evlFLTL}_4 : \Sigma \times \mathrm{LTL} \to \mathbb{B}_4 \times \mathrm{LTL}$ inductively as follows:

$$\mathrm{evlFLTL}_4(a, \mathrm{true}) = (\top, \mathrm{true})$$

$$\mathrm{evlFLTL}_4(a, \mathrm{false}) = (\bot, \mathrm{false})$$

$$\mathrm{evlFLTL}_4(a, p) = \begin{cases} (\top, \mathrm{true}) & \text{if } p \in a \\ (\bot, \mathrm{false}) & \text{else} \end{cases}$$

$$\mathrm{evlFLTL}_4(a, \neg p) = \begin{cases} (\bot, \mathrm{false}) & \text{if } p \in a \\ (\top, \mathrm{true}) & \text{else} \end{cases}$$

Targets & Outline
Truth Domains
Motivation
Definition
Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function
Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV
Conclusion

evlFLTL_4

Let $\Sigma = 2^{\text{AP}}$ be the finite alphabet, $p \in \text{AP}$ an atomic proposition, $a \in \Sigma$ a letter, and φ and ψ LTL formulae.

We then define the function $\text{evlFLTL}_4 : \Sigma \times \text{LTL} \to \mathbb{B}_4 \times \text{LTL}$ inductively as follows:

$$
\begin{aligned}
\text{evlFLTL}_4(a, \varphi \vee \psi) = (v_\varphi \sqcup v_\psi, \varphi' \vee \psi'), \text{ where} \\
(v_\varphi, \varphi') = \text{evlFLTL}_4(a, \varphi) \text{ and} \\
(v_\psi, \psi') = \text{evlFLTL}_4(a, \psi) \\
\text{evlFLTL}_4(a, \varphi \wedge \psi) = (v_\varphi \sqcap v_\psi, \varphi' \wedge \psi'), \text{ where} \\
(v_\varphi, \varphi') = \text{evlFLTL}_4(a, \varphi) \text{ and} \\
(v_\psi, \psi') = \text{evlFLTL}_4(a, \psi)
\end{aligned}
$$

Targets & Outline

Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

$\mathrm{evlFLTL}_4$

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $a \in \Sigma$ a letter, and φ and ψ LTL formulae.

We then define the function $\mathrm{evlFLTL}_4 : \Sigma \times \mathrm{LTL} \to \mathbb{B}_4 \times \mathrm{LTL}$ inductively as follows:

$$
\begin{aligned}
\mathrm{evlFLTL}_4(a, \mathrm{X}\,\varphi) &= (\bot^p, \varphi)\\
\mathrm{evlFLTL}_4(a, \overline{\mathrm{X}}\,\varphi) &= (\top^p, \varphi)\\
\mathrm{evlFLTL}_4(a, \varphi\,\mathrm{U}\,\psi) &= \mathrm{evlFLTL}_4(a, \psi \vee (\varphi \wedge \mathrm{X}(\varphi\,\mathrm{U}\,\psi)))\\
\mathrm{evlFLTL}_4(a, \varphi\,\mathrm{R}\,\psi) &= \mathrm{evlFLTL}_4(a, \psi \wedge (\varphi \vee \overline{\mathrm{X}}(\varphi\,\mathrm{R}\,\psi)))\\
\mathrm{evlFLTL}_4(a, \mathrm{F}\,\varphi) &= \mathrm{evlFLTL}_4(a, \varphi \vee \mathrm{X}\,\mathrm{F}\,\varphi)\\
\mathrm{evlFLTL}_4(a, \mathrm{G}\,\varphi) &= \mathrm{evlFLTL}_4(a, \varphi \wedge \overline{\mathrm{X}}\,\mathrm{G}\,\varphi)
\end{aligned}
$$

Examples

Example (Impartial Evaluation of Globally)

Consider $w = \{a\}\{a\}\emptyset$. First letter:

$$\begin{aligned}\mathrm{evlFLTL}_4(\{a\}, \mathrm{G}\, a) &= \mathrm{evlFLTL}_4(\{a\}, a \wedge \overline{\mathrm{X}}\, \mathrm{G}\, a) \\ &= (v_1 \sqcap v_2, \varphi_1 \wedge \varphi_2) \\ &= (\top \sqcap \top^p, \mathrm{true} \wedge \mathrm{G}\, a) \\ &= (\top^p, \mathrm{G}\, a)\end{aligned}$$

$$\begin{aligned}\text{where } (v_1, \varphi_1) &= \mathrm{evlFLTL}_4(\{a\}, a) = (\top, \mathrm{true}) \\ (v_2, \varphi_2) &= \mathrm{evlFLTL}_4(\{a\}, \overline{\mathrm{X}}\, \mathrm{G}\, a) = (\top^p, \mathrm{G}\, a).\end{aligned}$$

Next letters:

- $\mathrm{evlFLTL}_4(\{a\}, \mathrm{G}\, a) = (\top^p, \mathrm{G}\, a)$
- $\mathrm{evlFLTL}_4(\emptyset, \mathrm{G}\, a) = (\bot, \mathrm{false})$

Targets & Outline

Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

Examples

Example (Impartial Evaluation of Finally)

Consider $w = \emptyset\emptyset\{a\}$. First letter:

$$\begin{aligned} \mathrm{evlFLTL}_4(\emptyset, \mathrm{F}\, a) &= \mathrm{evlFLTL}_4(\emptyset, a \vee \mathrm{X}\,\mathrm{F}\, a) \\ &= (v_1 \sqcup v_2, \varphi_1 \vee \varphi_2) \\ &= (\bot \sqcup \bot^p, \mathrm{false} \vee \mathrm{F}\, a) \\ &= (\bot^p, \mathrm{F}\, a) \end{aligned}$$

where $(v_1, \varphi_1) = \mathrm{evlFLTL}_4(\emptyset, a) = (\bot, \mathrm{false})$
$(v_2, \varphi_2) = \mathrm{evlFLTL}_4(\emptyset, \mathrm{X}\,\mathrm{F}\, a) = (\bot^p, \mathrm{F}\, a)$.

Next letters:

- $\mathrm{evlFLTL}_4(\emptyset, \mathrm{F}\, a) = (\bot^p, \mathrm{F}\, a)$
- $\mathrm{evlFLTL}_4(\{a\}, \mathrm{F}\, a) = (\top, \mathrm{true})$

Section

Mealy Machines

Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Monitoring LTL on finite but expanding words

Automata-theoretic approach

- Synthesize automaton
- Monitoring = stepping through automaton

Targets & Outline

Truth Domains
Motivation
Definition

Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function

Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Conclusion

Finite-state Machines With Output

Moore Machines and Mealy Machines

- consists of states and transitions.
- read input and write output.
- change current state depending on input.

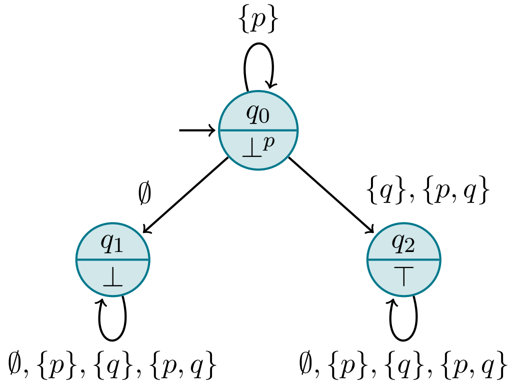
Moore Machines

- output depends only on current state.
- generate first output in initial state.

Mealy Machines

- output depends on current state and input.
- generate no output without input.

Moore Machine



Targets & Outline

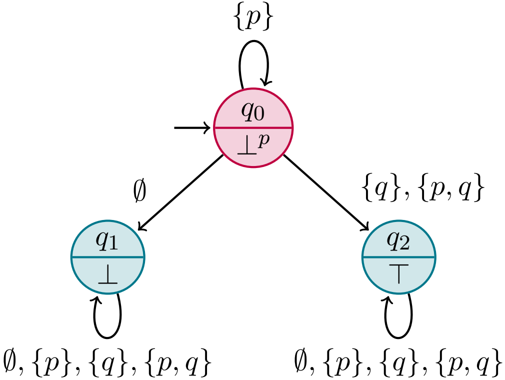
Truth Domains
Motivation
Definition

Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function

Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Conclusion

Moore Machine



Input

Output

$\bot^p$

Targets & Outline

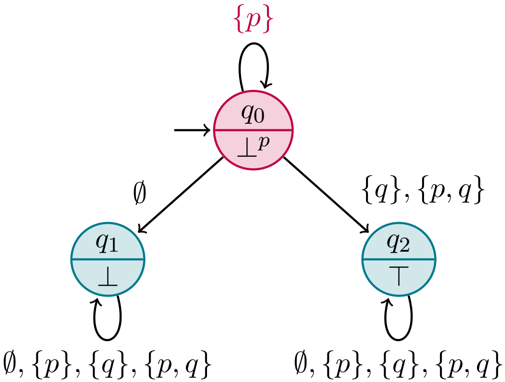
Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

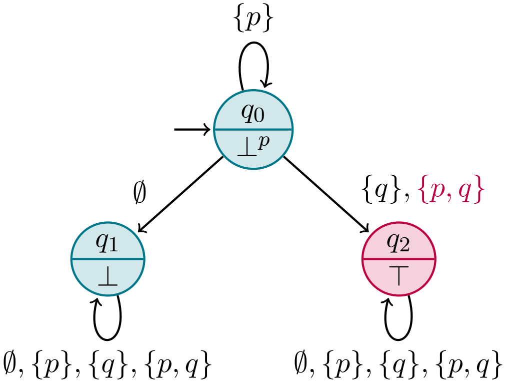
Conclusion

Moore Machine



Input	Output
$\{p\}$	$\bot^p \bot^p$

Moore Machine



Input

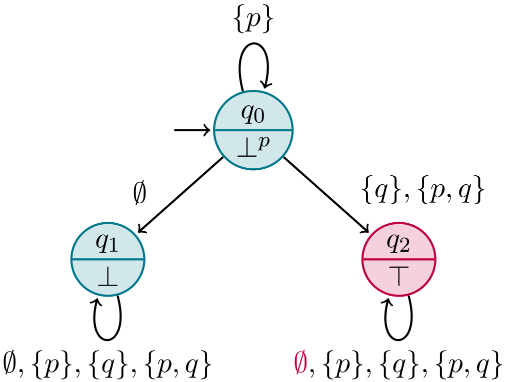
$\{p\}\{p, q\}$

Output

$\bot^p \bot^p \top$

Targets & Outline
Truth Domains
Motivation
Definition
Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function
Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV
Conclusion

Moore Machine



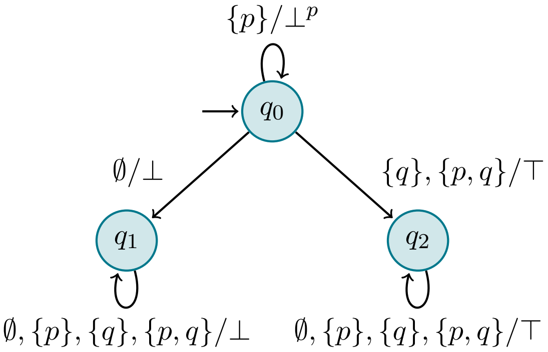
Input

$\{p\}\{p, q\}\emptyset$

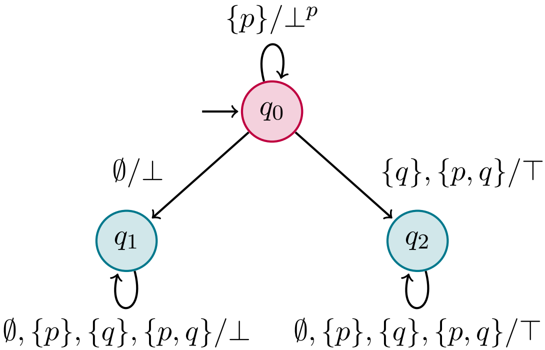
Output

$\bot^p \bot^p \top \top$

Mealy Machine



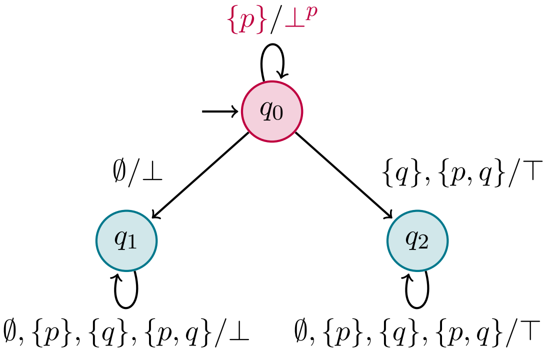
Mealy Machine



Input

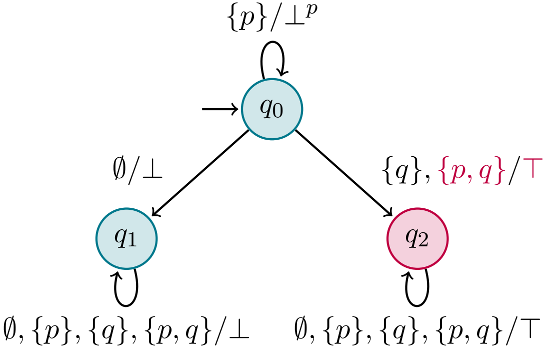
Output

Mealy Machine



Input	Output
$\{p\}$	$\bot^p$

Mealy Machine



Input	Output
$\{p\}\{p,q\}$	$\bot^p\top$

Targets & Outline

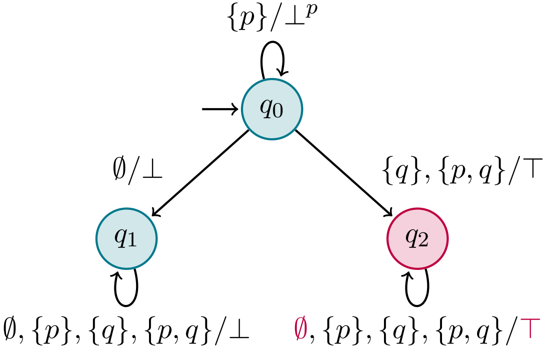
Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

Mealy Machine



Input

$\{p\}\{p,q\}\emptyset$

Output

$\bot^p\top\top$

Targets & Outline

Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

Translation

Translating Moore into Mealy Machine

- Keep states.
- Label transitions with output of target state.

Translating Mealy into Moore Machine

- Add some extra states for those with multiple incoming transitions labeled with different output.
- Label state with output from its incoming transitions.

From $\mathrm{evlFLTL}_4$ To Mealy Machine

- $\mathrm{evlFLTL}_4$ gets a letter and a formula and outputs a logic value and a new formula.
- Use formula as state of the Mealy Machine.
- Use letter as input and logic value as output.
- Next state (new formula) depends on state (formula) and input (letter).
- Output depends on state (formula) and input (letter).

Determinstic Mealy Machine

Definition (Deterministic Mealy Machine)

A (deterministic) *Mealy machine* is a tupel $\mathcal{M} = (\Sigma, Q, q_0, \Gamma, \delta)$ where

- Σ is the *input alphabet*,
- Q is a finite set of *states*,
- $q_0 \in Q$ is the *initial state*,
- Γ is the *output alphabet* and
- $\delta : Q \times \Sigma \to \Gamma \times Q$ is the *transition function*

Run of a Deterministic Mealy Machine

Definition (Run of a Deterministic Mealy Machine)

A *run* of a (deterministic) Mealy machine $\mathcal{M} = (\Sigma, Q, q_0, \Gamma, \delta)$ on a finite word $w \in \Sigma^n$ with outputs $o_i \in \Gamma$ is a sequence

$$t_0 \overset{(w_1,o_1)}{\rightarrow} t_1 \overset{(w_2,o_2)}{\rightarrow} \dots \overset{(w_{n-1},o_{n-1})}{\rightarrow} t_{n-1} \overset{(w_n,o_n)}{\rightarrow} t_n$$

such that

- $t_0 = q_0$ and
- $(t_i, o_i) = \delta(t_{i-1}, w_i)$

The *output* of the run is o_n.

Machine Types

Deterministic in one state at a time
transition function provides one state

Nondeterministic in one state at a time
transition function provides set of states

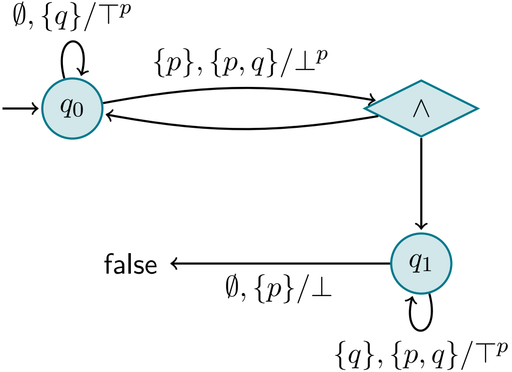
Universal in many states simultaneously
transition function provides set of states

Alternating in many states simultaneously
transition function provides positive Boolean combination of states

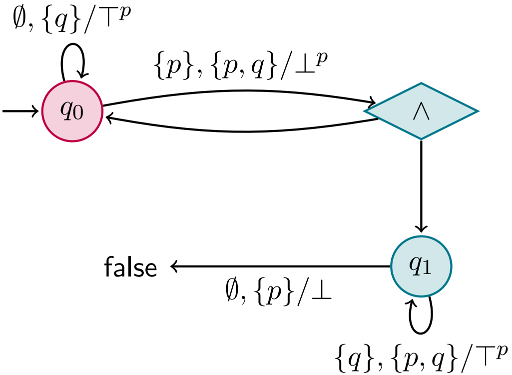
FLTL$_4$ Monitoring Needs Alternating Mealy Machines

We need all positive Boolean combinations
of subformulae with finitely many states.

Alternating Mealy Machine



Alternating Mealy Machine



Input	Output

Targets & Outline

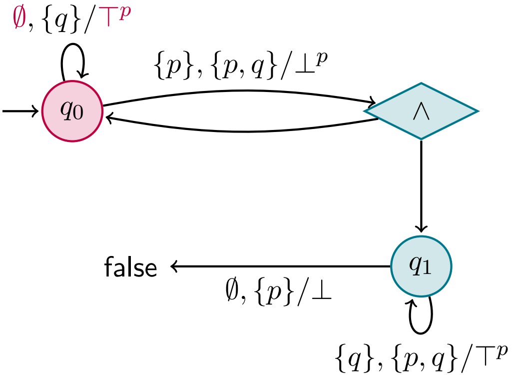
Truth Domains
Motivation
Definition

Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function

Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Conclusion

Alternating Mealy Machine



Input	Output
$\emptyset$	$\top^p$

Targets & Outline

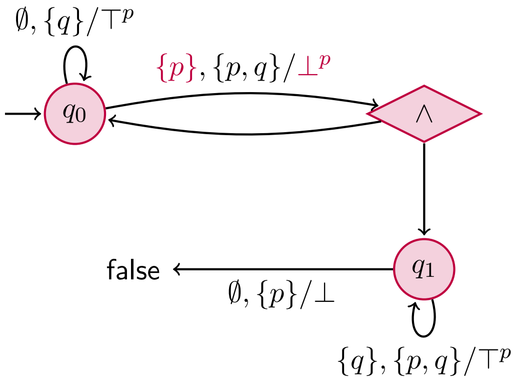
Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

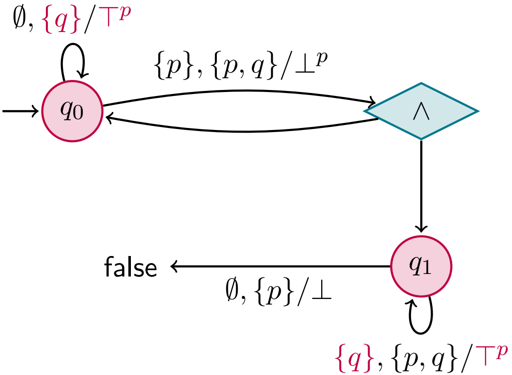
Alternating Mealy Machine



Input	Output
$\emptyset\{p\}$	$\top^p\bot^p$

Targets & Outline
Truth Domains
Motivation
Definition
Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function
Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV
Conclusion

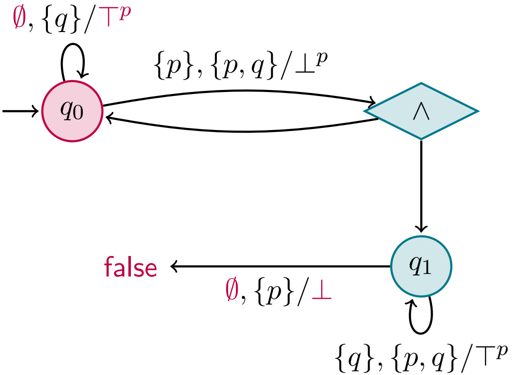
Alternating Mealy Machine



Input
$\emptyset\{p\}\{q\}$

Output
$\top^p\bot^p\top^p$

Alternating Mealy Machine



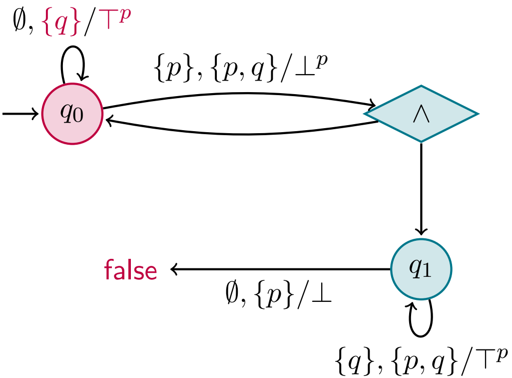
Input

$\emptyset\{p\}\{q\}\emptyset$

Output

$\top^p\bot^p\top^p\bot$

Alternating Mealy Machine



Input

$\emptyset\{p\}\{q\}\emptyset\{q\}$

Output

$\top^p\bot^p\top^p\bot\bot$

Positive Boolean Combination (PBC)

Definition (Positive Boolean Combination (PBC))

Given a set Q we define the set of all *positive Boolean combinations* (PBC) over Q, denoted by $B^+(Q)$, inductively as follows:

- $\{\text{true}, \text{false}\} \subseteq B^+(Q)$,
- $Q \subseteq B^+(Q)$ and
- $\forall \alpha, \beta \in B^+(Q) : \alpha \vee \beta, \alpha \wedge \beta \in B^+(Q)$.

Examples

Consider $\text{AP} = \{a, b, c\}$

- $a \in B^+(\text{AP}), \{a\} \notin B^+(\text{AP})$,
- $a \wedge b \vee a \wedge c \in B^+(\text{AP})$,
- $\text{true} \in B^+(\text{AP})$ and $\text{false} \in B^+(\text{AP})$.

Alternating Mealy Machine (AMM)

Definition (Alternating Mealy Machine (AMM))

A *alternating Mealy machine* (AMM) is a tupel $\mathcal{M} = (\Sigma, Q, q_0, \Gamma, \delta)$ where

- Σ is the *input alphabet*,
- Q is a finite set of *states*,
- $q_0 \in Q$ is the *initial state* and
- Γ is a finite, distributive lattice, the *output lattice*,
- $\delta : Q \times \Sigma \to B^+(\Gamma \times Q)$ is the *transition function*

Alternating Mealy Machine (AMM)

Definition (Alternating Mealy Machine (AMM))

A *alternating Mealy machine* (AMM) is a tupel $\mathcal{M} = (\Sigma, Q, q_0, \Gamma, \delta)$ where

- Σ is the *input alphabet*,
- Q is a finite set of *states*,
- $q_0 \in Q$ is the *initial state* and
- Γ is a finite, distributive lattice, the *output lattice*,
- $\delta : Q \times \Sigma \to B^+(\Gamma \times Q)$ is the *transition function*

Convention

Understand $\delta : Q \times \Sigma \to B^+(\Gamma \times Q)$ as a function $\delta : Q \times \Sigma \to \Gamma \times B^+(Q)$

Extended Transition Function

Definition (Extended Transition Function)

Let $\delta : Q \times \Sigma \to \Gamma \times B^+(Q)$ be the transition function of an alternating mealy machine. Then the extended transition function $\hat{\delta} : B^+(Q) \times \Sigma \to \Gamma \times B^+(Q)$ is inductively defined as follows

- $\hat{\delta}(q, a) = \delta(q, a)$,
- $\hat{\delta}(\text{true}, a) = (\top, \text{true})$, $\quad \hat{\delta}(\text{false}, a) = (\bot, \text{false})$,
- $\hat{\delta}(q_1 \vee q_2, a) = (o_1 \sqcup o_2, q'_1 \vee q'_2)$ and
- $\hat{\delta}(q_1 \wedge q_2, a) = (o_1 \sqcap o_2, q'_1 \wedge q'_2)$,
  where $(o_1, q'_1) = \hat{\delta}(q_1, a)$ and $(o_2, q'_2) = \hat{\delta}(q_2, a)$.

Run of an Alternating Mealy Machine

Definition (Run of an Alternating Mealy Machine)

A *run* of an alternating Mealy machine $\mathcal{M} = (\Sigma, Q, q_0, \Gamma, \delta)$ on a finite word $w \in \Sigma^n$ with outputs $o_i \in \Gamma$ is a sequence

$$t_0 \overset{(w_1,o_1)}{\rightarrow} t_1 \overset{(w_2,o_2)}{\rightarrow} \dots \overset{(w_{n-1},o_{n-1})}{\rightarrow} t_{n-1} \overset{(w_n,o_n)}{\rightarrow} t_n$$

such that

$$t_0 = q_0 \text{ and } (t_i, o_i) = \hat{\delta}(t_{i-1}, w_i),$$

where $\hat{\delta}$ is the extended transition function of $\mathcal{M}$.
The *output* of the run is o_n.

Equivalence of PBCs

Definition (Model Relation for PBCs)

Let Q be a set. A subset $S \subseteq Q$ is a *model* of a positive Boolean combination $\alpha \in B^+(Q)$, denoted by $S \models \alpha$, iff α evaluates to true in propositional logic interpreting all $p \in S$ as true and all $p \in Q \backslash S$ as false.

Definition (Equivalence of PBCs)

Let Q be a set and $\alpha \in B^+(Q)$ and $\beta \in B^+(Q)$ be positive Boolean combinations over Q. α and β are *equivalent*, denoted by $\alpha \equiv \beta$, iff

$$\forall S \subseteq Q : S \models \alpha \Leftrightarrow S \models \beta.$$

Targets & Outline

Truth Domains
- Motivation
- Definition

Four-Valued LTL Semantics: FLTL_4
- Definition
- Monitor Function

Mealy Machines
- Determinstic Mealy Machines
- Alternating Mealy Machines
- Automata Based RV

Conclusion

Equivalence Classes of PBCs

Definition (Equivalence Classes of PBCs)

Let Q be a set. The *equivalence class* $[\alpha]$ of a positive Boolean combination $\alpha \in B^+(Q)$ over Q is defined as follows

$$[\alpha] = \{\beta \in B^+(Q) \mid \alpha \equiv \beta\}.$$

The set of all equivalence classes of positive Boolean combinations over Q is denoted by the following *quotient set*

$$B^+(Q)/\!\equiv\; = \{[\alpha] \mid \alpha \in B^+(Q)\}.$$

Translating AMM to MM

- Use $B^+(Q)$ instead of Q as states.
- The extended transition function $\hat{\delta}$ then can be used as transition function of a MM.
- Problem: $B^+(Q)$ is infinite.
- There are infinitely many positive Boolean combinations over any set with at least two elements.
- Solution: Use $B^+(Q)/\equiv$ instead of $B^+(Q)$.

Equivalence of Output and Next State

Lemma

Let $\hat{\delta}$ *be the extended transition function of an AMM* $\mathcal{M} = (\Sigma, Q, q_0, \Gamma, \delta)$, $a \in \Sigma$, $o, p \in \Gamma$ *and* $\alpha, \beta, \alpha', \beta' \in B^+(Q)$ *such that*

$$\alpha \equiv \beta,$$
$$(o, \alpha') = \hat{\delta}(\alpha, a) \text{ and}$$
$$(p, \beta') = \hat{\delta}(\beta, a).$$

Then

$$o = p \text{ and} \qquad (*)$$
$$\alpha' \equiv \beta'.$$

The proof of $(*)$ requires the output lattice to be distributive.

Representatives of $[q]$

- We can now use $B^+(Q)/\equiv$ instead of Q as states.
- We still need a well defined representative for $[\alpha]$ for $\alpha \in B^+(Q)$.
- In other words: Given $\alpha \in Q$, howto find $[\alpha]$?
- Solution: Use disjunctive normal form of α.

Targets & Outline

Truth Domains
Motivation
Definition

Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function

Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Conclusion

Disjunctive Normal Form (DNF)

Definition (Disjunctive Normal Form (DNF))

A positive Boolean combination $\alpha \in B^+(Q)$ over a set Q is in *disjunctive normal form* (DNF) iff

$$\alpha = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} q_{i,j}$$

for $q_{i,j} \in Q$. A disjunctive normal form $\alpha \in B^+(Q)$ is called *minimal* if there is no disjunctive normal form $\beta \in B^+(Q)$ s. t. $\alpha \equiv \beta$ and β contains less operators.

Disjunctive Normal Form (DNF)

Definition (Disjunctive Normal Form (DNF))

A positive Boolean combination $\alpha \in B^+(Q)$ over a set Q is in *disjunctive normal form* (DNF) iff

$$\alpha = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} q_{i,j}$$

for $q_{i,j} \in Q$. A disjunctive normal form $\alpha \in B^+(Q)$ is called *minimal* if there is no disjunctive normal form $\beta \in B^+(Q)$ s. t. $\alpha \equiv \beta$ and β contains less operators.

Lemma

For every positive Boolean combination $\alpha \in B^+(Q)$ there exists a positive Boolean combination β such that $\alpha \equiv \beta$ and β is in minimal DNF.

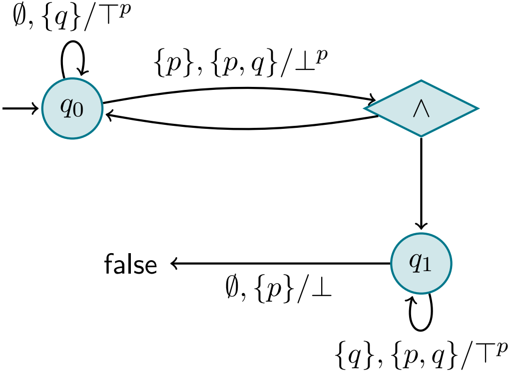
Proof uses distributivity of propositional logic.

$B^+(Q)/\equiv$ is finite

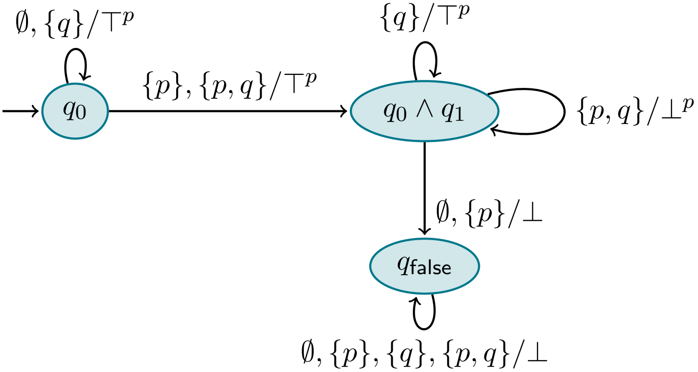
- Let Q be the set of states of an AMM.
- Then Q is finite.
- Then there are at most $2^{|Q|}$ many different α for
$$\alpha = \bigvee_{i=1}^{n} q_i \text{ and } n \text{ different } q_i \in Q.$$
- Then there are at most $2^{2^{|Q|}}$ many different β for
$$\beta = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} q_{i,j} \text{ minimal and } q_{i,j} \in Q.$$
- Then there are at most $2^{2^{|Q|}}$ many different $[\beta]$ for $\beta \in B^+(Q)$.

Targets & Outline
Truth Domains
Motivation
Definition
Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function
Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV
Conclusion

Example: Alternating Mealy Machine



Example: Translated Mealy Machine



Translating AMM to Non-Deterministic or Universal MM

Analogous we can use sets of (not equivalent) disjunctions or conjunctions of Q as states instead of $B^+(Q)/\equiv$.

- alternating $\rightarrow$ non-deterministic: translate $t \in B^+(Q)$ into equivalent minimal disjunctive normal form and use monomials as new states.
- alternating $\rightarrow$ universal: translate $t \in B^+(Q)$ into equivalent minimal conjunctive normal form and use clauses as new states.
- alternating $\rightarrow$ deterministic: translate $t \in B^+(Q)$ into equivalent minimal conjunctive or disjunctive normal form and use normal forms as new states.

Automata Based RV

We monitor an LTL formula φ by evaluating its current subformula ψ w.r.t. the current letter a.

The monitor function $\mathrm{evlFLTL}_4$, which

- takes an LTL formula ψ in NNF and a letter $a \in \Sigma$ and
- returns $[\![a \models \psi]\!]_4$ and a new LTL formula ψ',

can be interpreted as transition function of an AMM where

- the states are subformulae of φ,
- the initial state is φ,
- the current state is ψ,
- we read the letter a,
- we output $[\![a \models \psi]\!]_4$ and
- the next state is the new formula ψ'.

Transition function of an AMM

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $a \in \Sigma$ a letter, φ, ψ_1, ψ_2 LTL formulae in NNF and Q the set of all subformulae of φ.

We then define the transition function $\delta_4^a : Q \times \Sigma \to B^+(\mathbb{B}_4 \times Q)$ of the monitor AMM $\mathcal{M}_\varphi = (\Sigma, Q, \varphi, \mathbb{B}_4, \delta_4^a)$ inductively as follows:

$$\delta_4^a(\mathrm{true}, a) = (\top, \mathrm{true})$$

$$\delta_4^a(\mathrm{false}, a) = (\bot, \mathrm{false})$$

$$\delta_4^a(p, a) = \begin{cases} (\top, \mathrm{true}) & \text{if } p \in a \\ (\bot, \mathrm{false}) & \text{else} \end{cases}$$

$$\delta_4^a(\neg p, a) = \begin{cases} (\top, \mathrm{true}) & \text{if } p \notin a \\ (\bot, \mathrm{false}) & \text{else} \end{cases}$$

Transition function of an AMM

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $a \in \Sigma$ a letter, φ, ψ_1, ψ_2 LTL formulae in NNF and Q the set of all subformulae of φ.

We then define the transition function $\delta_4^a : Q \times \Sigma \to B^+(\mathbb{B}_4 \times Q)$ of the monitor AMM $\mathcal{M}_\varphi = (\Sigma, Q, \varphi, \mathbb{B}_4, \delta_4^a)$ inductively as follows:

$$
\begin{aligned}
\delta_4^a(\psi_1 \vee \psi_2, a) &= \delta_4^a(\psi_1, a) \vee \delta_4^a(\psi_2, a) \\
\delta_4^a(\psi_1 \wedge \psi_2, a) &= \delta_4^a(\psi_1, a) \wedge \delta_4^a(\psi_2, a) \\
\delta_4^a(\mathrm{X}\, \psi_1, a) &= (\bot^p, \psi_1) \\
\delta_4^a(\overline{\mathrm{X}}\, \psi_1, a) &= (\top^p, \psi_1)
\end{aligned}
$$

Transition function of an AMM

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $a \in \Sigma$ a letter, φ, ψ_1, ψ_2 LTL formulae in NNF and Q the set of all subformulae of φ.

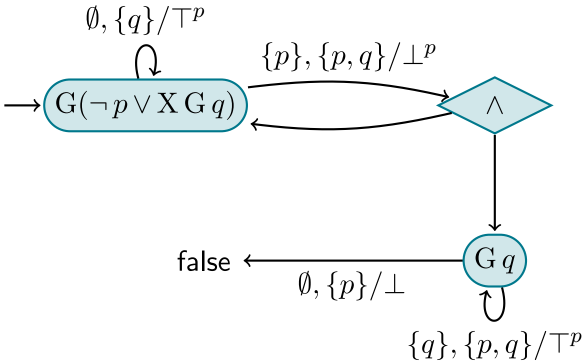
We then define the transition function $\delta_4^a : Q \times \Sigma \to B^+(\mathbb{B}_4 \times Q)$ of the monitor AMM $\mathcal{M}_\varphi = (\Sigma, Q, \varphi, \mathbb{B}_4, \delta_4^a)$ inductively as follows:

$$\delta_4^a(\psi_1 \,\mathrm{U}\, \psi_2, a) = \delta_4^a(\psi_2 \vee (\psi_1 \wedge \mathrm{X}(\psi_1 \,\mathrm{U}\, \psi_2)), a)$$
$$\delta_4^a(\psi_1 \,\mathrm{R}\, \psi_2, a) = \delta_4^a(\psi_2 \wedge (\psi_1 \vee \overline{\mathrm{X}}(\psi_1 \,\mathrm{R}\, \psi_2)), a)$$
$$\delta_4^a(\mathrm{F}\, \psi_1, a) = \delta_4^a(\psi_1 \vee (\mathrm{X}(\mathrm{F}\, \psi_1)), a)$$
$$\delta_4^a(\mathrm{G}\, \psi_1, a) = \delta_4^a(\psi_1 \wedge (\overline{\mathrm{X}}(\mathrm{G}\, \psi_1)), a)$$

Targets & Outline
Truth Domains
Motivation
Definition
Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function
Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV
Conclusion

Example

Graph of the monitor $\mathcal{M}_\varphi$ of the formula $\varphi = \mathrm{G}(p \rightarrow \mathrm{X}\,\mathrm{G}\,q)$:



Targets & Outline

Truth Domains

Motivation

Definition

Four-Valued LTL Semantics: FLTL_4

Definition

Monitor Function

Mealy Machines

Determinstic Mealy Machines

Alternating Mealy Machines

Automata Based RV

Conclusion

Generating Deterministic Monitors

In practical implementations one may omit the AMM and generate the MM directly out of the LTL formula.

- Define a function $\mathrm{smplfy} : \mathrm{LTL} \to \mathrm{LTL}$ that transforms LTL formulae into a unique normal form.
- Use all simplified positive Boolean combinations of subformulae of φ as states for $\mathcal{M}_\varphi$.
- Define $\delta_4 : Q \times \Sigma \to \mathbb{B}_4 \times Q$ inductively as follows:

$$\delta_4(\psi_1 \vee \psi_2, a) = (v_{\psi_1} \sqcup v_{\psi_2}, \mathrm{smplfy}(\psi_1' \vee \psi_2')), \text{ where}$$
$$(v_{\psi_1}, \psi_1') = \delta_4(\psi_1, a) \text{ and}$$
$$(v_{\psi_2}, \psi_2') = \delta_4(\psi_2, a)$$
$$\delta_4(\psi_1 \wedge \psi_2, a) = (v_{\psi_1} \sqcap v_{\psi_2}, \mathrm{smplfy}(\psi_1' \wedge \psi_2')), \text{ where}$$
$$(v_{\psi_1}, \psi_1') = \delta_4(\psi_1, a) \text{ and}$$
$$(v_{\psi_2}, \psi_2') = \delta_4(\psi_2, a)$$
$$\delta_4(\psi_1, a) = \delta_4^a(\psi_1, a) \text{ for any other formula } \psi_1.$$

Automata-theoretic Approach

Further Topics

- Alternating vs. non-deterministic vs. deterministic machines.

Targets & Outline

Truth Domains
Motivation
Definition

Four-Valued LTL Semantics: FLTL_4
Definition
Monitor Function

Mealy Machines
Determinstic Mealy Machines
Alternating Mealy Machines
Automata Based RV

Conclusion

Automata-theoretic Approach

Further Topics

- Alternating vs. non-deterministic vs. deterministic machines.
- Complexity of the translations.

Automata-theoretic Approach

Further Topics

- Alternating vs. non-deterministic vs. deterministic machines.
- Complexity of the translations.
- Size vs. power.

Automata-theoretic Approach

Further Topics

- Alternating vs. non-deterministic vs. deterministic machines.
- Complexity of the translations.
- Size vs. power.
- State sequence for an input word.

Conclusion

1. Every two-valued logic is not impartial. Impartiality implies multiple values.
2. We use the distributive De Morgan lattice $\mathbb{B}_4 = \{\bot, \bot^p, \top^p, \top\}$ in the impartial FLTL_4 semantics.
3. At the end of the word X evaluates to $\bot^p$ and $\overline{\mathrm{X}}$ evaluates to $\top^p$.
4. $\mathrm{evlFLTL}_4$ performs formula rewriting (progression) for an LTL formula and one letter.
5. $\mathrm{evlFLTL}_4$ can be used to describe the transition function of an alternating mealy machine using the subformulae as states.
6. Such an alternating mealy machine can be translated into a deterministic mealy machine using the fact that equivalent positive combinations of states leads to the same next states and output.

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Chapter 5

Anticipatory LTL Semantics

Course "Runtime Verification"
M. Leucker & V. Stolz
INF5140 / V17

Chapter 5

Learning Targets of Chapter “Anticipatory LTL Semantics”.

1. Understand that LTL semantics can be defined over infinite words as well.
2. Understand the difference of LTL over finite and infinite words.
3. Recall anticipation and understand why impartiality is not enough to build good monitors.
4. Get used to the three-valued sematics for LTL.
5. Understand the concept of safety and co-safety properties and get an idea of monitorable properties.

Chapter 5

Outline of Chapter "Anticipatory LTL Semantics".

LTL on Infinite Words

- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$

- Anticipation
- Definition
- Examples

Monitorable Properties

- (Co-)Safety
- Examples
- Monitorability

Section

LTL on Infinite Words

Semantics

Equivalences and Examples

Need of LTL on Infinite Words in RV?

Impartiality Say $\top$ or $\bot$ only if you are sure.

Anticipation Say $\top$ or $\bot$ once you can be sure.

We want do define impartial LTL semantics:

- Say $\top$ if every infinite continuation evaluates to $\top$.
- Say $\bot$ if every infinite continuation evaluates to $\bot$.
- Otherwise say $?$.

We Need LTL on Infinite Words

- Impartial LTL semantics will be based on infinite continuations.
- Properties of infinite continuations cannot be expressed using LTL on finite words.

Infinite Words

- An infinite word w is an infinite sequence over the alphabet $\Sigma = 2^{\mathrm{AP}}$.
- w can be interpreted as function $w : \mathbb{N} \backslash \{0\} \to \Sigma$.
- w can be interpreted as concatenation of many finite and one infinite words.

Examples (Infinite Words)

Consider the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}^\omega$ denotes the infinite word where every letter is $\{p\}$ and can be interpreted as $w(i) = \{p\}$ for all $i \geq 1$.
- $\emptyset(\{q\}\{p\})^\omega$ can be interpreted as

$$w(i) = \begin{cases} \emptyset & \text{if } i = 1 \\ \{q\} & \text{if } i \equiv 0 \mod 2 \\ \{p\} & \text{else} \end{cases}$$

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Finally and Globally Examples

Examples (Finally and Globally)

Consider infinite words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset^\omega \models \mathrm{F}\, q$.

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Finally and Globally Examples

Examples (Finally and Globally)

Consider infinite words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset^\omega \models \mathrm{F}\, q$.
- $\{q\}\{q\}(\{p,q\}\{q\})^\omega \models \mathrm{G}\, q$.

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Finally and Globally Examples

Examples (Finally and Globally)

Consider infinite words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset^\omega \models \mathrm{F}\, q$.
- $\{q\}\{q\}(\{p, q\}\{q\})^\omega \models \mathrm{G}\, q$.
- $(\emptyset\{p\}\{p, q\}\emptyset)^\omega \models \mathrm{G}\,\mathrm{F}\, q$.

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

Finally and Globally Examples

Examples (Finally and Globally)

Consider infinite words over the alphabet $\Sigma = 2^{\mathrm{AP}}$ with $\mathrm{AP} = \{p, q\}$.

- $\{p\}\emptyset\{q\}\emptyset^\omega \models \mathrm{F}\, q$.
- $\{q\}\{q\}(\{p,q\}\{q\})^\omega \models \mathrm{G}\, q$.
- $(\emptyset\{p\}\{p,q\}\emptyset)^\omega \models \mathrm{G}\,\mathrm{F}\, q$.
- $\{p\}\emptyset\{q\}\{p\}(\{p,q\}\{q\})^\omega \models \mathrm{F}\,\mathrm{G}\, q$.

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Parts of Infinite Words

In the formal definition of LTL semantics we denote parts of a word as follows:

Let $w = a_1 a_2 a_3 \ldots \in \Sigma^\omega$ be an infinite word over the alphabet $\Sigma = 2^{\mathrm{AP}}$ and let $i \in \mathbb{N}$ with $i \geq 1$ be a position in this word. Then

- $w_i = a_i$ is the i-th letter of the word,
- $w^{(i)} = a_1 a_2 \ldots a_i$ is the prefix of w of length i and
- w^i is the subword of w s. t. $w = w^{(i-1)} w^i$.

LTL Semantics on Infinite Words

Definition (LTL Semantics on Infinite Words)

Let φ, ψ be LTL formulae and let $w \in \Sigma^\omega$ be an infinite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$
\begin{array}{ll}
w \models \text{true} & \\
w \models p & \text{iff } p \in w_1 \\
w \models \neg p & \text{iff } p \notin w_1 \\
w \models \neg \varphi & \text{iff } w \not\models \varphi \\
w \models \varphi \vee \psi & \text{iff } w \models \varphi \text{ or } w \models \psi \\
w \models \varphi \wedge \psi & \text{iff } w \models \varphi \text{ and } w \models \psi
\end{array}
$$

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

LTL Semantics on Infinite Words

Definition (LTL Semantics on Infinite Words)

Let φ, ψ be LTL formulae and let $w \in \Sigma^\omega$ be an infinite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$\begin{array}{ll} w \models \mathrm{X}\,\varphi & \text{iff } w^2 \models \varphi \\ w \models \overline{\mathrm{X}}\,\varphi & \text{iff } w^2 \models \varphi \end{array}$$

LTL Semantics on Infinite Words

Definition (LTL Semantics on Infinite Words)

Let φ, ψ be LTL formulae and let $w \in \Sigma^\omega$ be an infinite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$\begin{aligned} w \models \varphi \,\mathrm{U}\, \psi \quad & \text{iff } \exists i \geq 1 : (w^i \models \psi \\ & \quad \text{and } \forall k, 1 \leq k < i : w^k \models \varphi) \\ w \models \varphi \,\mathrm{R}\, \psi \quad & \text{iff } \exists i \geq 1 : (w^i \models \varphi \\ & \quad \text{and } \forall k, 1 \leq k \leq i : w^k \models \psi) \\ & \text{or } \forall i \geq 1 : w^i \models \psi \end{aligned}$$

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL_3

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

LTL Semantics on Infinite Words

Definition (LTL Semantics on Infinite Words)

Let φ, ψ be LTL formulae and let $w \in \Sigma^\omega$ be an infinite word. Then the semantics of φ with respect to w is inductively defined as follows:

$$w \models \mathrm{F}\,\varphi \quad \text{iff } \exists i \geq 1 : w^i \models \varphi$$

$$w \models \mathrm{G}\,\varphi \quad \text{iff } \forall i \geq 1 : w^i \models \varphi$$

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Semantic Function for LTL on Infinite Words

- We defined the LTL semantics on infinite words as relation $w \models \varphi$ between a word $w \in \Sigma^\omega$ and a LTL formula φ.
- This can be interpreted as semantic function

$$\mathrm{sem}_\omega : \Sigma^\omega \times \mathrm{LTL} \to \mathbb{B}_2,$$

$$\mathrm{sem}_\omega(w, \varphi) = [\![w \models \varphi]\!]_\omega := \begin{cases} \top & \text{if } w \models \varphi \\ \bot & \text{else.} \end{cases}$$

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Languages Defined by LTL Formulae

The set of models of an LTL formula φ defines a language $\mathcal{L}(\varphi) \subseteq \Sigma^\omega$ of infinite words over $\Sigma = 2^{\mathrm{AP}}$ as follows:

$$\mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid [\![w \models \varphi]\!]_\omega = \top\}.$$

Equivalences

Weak Next

Let $w \in \Sigma^\omega$ be an infinite word.

- w has no last character.
- For every position $i \geq 1$ the word $w^i \in \Sigma^\omega$ is infinite.
- X and $\overline{\mathrm{X}}$ have the same semantics.

The De Morgan rules, equivalences for G and F and the fixed point equations for U and R are still valid.

Finite and Infinite Semantics

Examples (Globally and Finally)

- We know

$$[\![\{p\}(\{p\}\{q\})^\omega \models \mathrm{F}\, q]\!]_\omega = \top$$

from

$$[\![\{p\}\{p\}\{q\} \models \mathrm{F}\, q]\!]_4 = \top.$$

- We know

$$[\![\{p\}(\{p\}\{q\})^\omega \models \mathrm{G}\, p]\!]_\omega = \bot$$

from

$$[\![\{p\}\{p\}\{q\} \models \mathrm{G}\, p]\!]_4 = \bot.$$

Finite and Infinite Semantics

Examples (Until)

- We know

$$[\![\{p\}(\{p\}\{q\})^\omega \models p \,\mathrm{U}\, q]\!]_\omega = \top$$

from

$$[\![\{p\}\{p\}\{q\} \models p \,\mathrm{U}\, q]\!]_4 = \top.$$

- We know

$$[\![\{p\}\emptyset\{q\}^\omega \models p \,\mathrm{U}\, q]\!]_\omega = \bot$$

from

$$[\![\{p\}\emptyset \models p \,\mathrm{U}\, q]\!]_4 = \bot.$$

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Section

Anticipatory LTL Semantics: LTL_3

- Anticipation
- Definition
- Examples

Chapter 5 “Anticipatory LTL Semantics”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

Anticipation

Be Anticipatory

- go for a final verdict ($\top$ or $\bot$) once you really know
- do not delay the decision

Definition (Anticipation)

Anticipation requires that once every (possibly infinite) continuation of a finite trace leads to the same verdict, then the finite trace evaluates to this very same verdict.

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

FLTL_4 is Not Anticipatory

Example (Next Operator)

We have

$$[\![\{p\} \models \mathrm{X}\,\mathrm{X}\,\mathrm{false}]\!]_4 = \bot^p$$

$$\begin{aligned}&[\![\{p\}\{p\} \models \mathrm{X}\,\mathrm{X}\,\mathrm{false}]\!]_4 \\ &= [\![\{p\} \models \mathrm{X}\,\mathrm{false}]\!]_4 = \bot^p\end{aligned}$$

$$\begin{aligned}&[\![\{p\}\{p\}\{p\} \models \mathrm{X}\,\mathrm{X}\,\mathrm{false}]\!]_4 \\ &= [\![\{p\}\{p\} \models \mathrm{X}\,\mathrm{false}]\!]_4 \\ &= [\![\{p\} \models \mathrm{false}]\!]_4 = \bot,\end{aligned}$$

but it would be anticipatory to have

$$[\![\{p\} \models \mathrm{X}\,\mathrm{X}\,\mathrm{false}]\!]_3 = \bot.$$

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

$\textsf{FLTL}_4$ is Not Anticipatory

Example (Globally and Finally Operator)

We have

$$[\![w \models \mathrm{G}\,\mathrm{true}]\!]_4 = [\![w \models \mathrm{false}\,\mathrm{R}\,\mathrm{true}]\!]_4 = \top^p \text{ and}$$
$$[\![w \models \mathrm{F}\,\mathrm{false}]\!]_4 = [\![w \models \mathrm{true}\,\mathrm{U}\,\mathrm{false}]\!]_4 = \bot^p$$

but it would be anticipatory to have

$$[\![w \models \mathrm{G}\,\mathrm{true}]\!]_3 = \top \text{ and}$$
$$[\![w \models \mathrm{F}\,\mathrm{false}]\!]_3 = \bot.$$

Impartial Anticipation

- Define LTL semantics for finite, non-terminated words.
- The set of all infinite continuations of a finite word contains only infinite words.
- Define semantics for finite words based on semantics of these infinite continuations.
- If the semantic function yields the same verdict for all infinite continuations use that verdict.
- Combine $\top^p$ and $\bot^p$ to a common $?$ for the other cases.

Anticipatory Three-Valued LTL Semantics

Definition (LTL$_3$ Semantics)

Let φ be an LTL formula and let $u \in \Sigma^*$ be a finite word. Then the semantics of φ with respect to u is defined as follows:

$$[\![u \models \varphi]\!]_3 = \begin{cases} \top & \text{if } \forall w \in \Sigma^\omega : [\![uw \models \varphi]\!]_\omega = \top \\ \bot & \text{if } \forall w \in \Sigma^\omega : [\![uw \models \varphi]\!]_\omega = \bot \\ ? & \text{else.} \end{cases}$$

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

Example

Consider $\varphi = \mathrm{G}(p \rightarrow \mathrm{F}\, false)$ and $\emptyset\{q\}\{p\}\emptyset \in \Sigma^*$ for $\Sigma = 2^{\mathrm{AP}}$ and $\mathrm{AP} = \{p, q\}$. We then have

- $[\![\emptyset \models \varphi]\!]_3 = ?$
- $[\![\emptyset\{q\} \models \varphi]\!]_3 = ?$
- $[\![\emptyset\{q\}\{p\} \models \varphi]\!]_3 = \bot$
- $[\![\emptyset\{q\}\{p\}\emptyset \models \varphi]\!]_3 = \bot$

Example

Consider $\varphi = \mathrm{G}(p \rightarrow \mathrm{F}\, false)$ and $\emptyset\{q\}\{p\}\emptyset \in \Sigma^*$ for $\Sigma = 2^{\mathrm{AP}}$ and $\mathrm{AP} = \{p, q\}$. We then have

- $[\![\emptyset \models \varphi]\!]_3 =?$
- $[\![\emptyset\{q\} \models \varphi]\!]_3 =?$
- $[\![\emptyset\{q\}\{p\} \models \varphi]\!]_3 = \bot$
- $[\![\emptyset\{q\}\{p\}\emptyset \models \varphi]\!]_3 = \bot$
- $[\![\emptyset\{q\}\{p\}u \models \varphi]\!]_3 = \bot$ for all $u \in \Sigma^*$

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Possible Verdicts of LTL Formulae

Consider a word $w \in \Sigma^*$ for $\Sigma = 2^{\mathrm{AP}}$ and propositions $p, q \in \mathrm{AP}$. We then have

- $[\![w \models p \,\mathrm{U}\, q]\!]_3 \in \{\top, ?, \bot\}$
- $[\![w \models p \,\mathrm{R}\, q]\!]_3 \in \{\top, ?, \bot\}$

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Possible Verdicts of LTL Formulae

Consider a word $w \in \Sigma^*$ for $\Sigma = 2^{\mathrm{AP}}$ and propositions $p, q \in \mathrm{AP}$. We then have

- $[\![w \models p \,\mathrm{U}\, q]\!]_3 \in \{\top, ?, \bot\}$
- $[\![w \models p \,\mathrm{R}\, q]\!]_3 \in \{\top, ?, \bot\}$
- $[\![w \models \mathrm{F}\, p]\!]_3 \in \{\top, ?\}$
- $[\![w \models \mathrm{G}\, p]\!]_3 \in \{?, \bot\}$

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Possible Verdicts of LTL Formulae

Consider a word $w \in \Sigma^*$ for $\Sigma = 2^{\mathrm{AP}}$ and propositions $p, q \in \mathrm{AP}$. We then have

- $[\![w \models p \,\mathrm{U}\, q]\!]_3 \in \{\top, ?, \bot\}$
- $[\![w \models p \,\mathrm{R}\, q]\!]_3 \in \{\top, ?, \bot\}$
- $[\![w \models \mathrm{F}\, p]\!]_3 \in \{\top, ?\}$
- $[\![w \models \mathrm{G}\, p]\!]_3 \in \{?, \bot\}$
- $[\![w \models \mathrm{G}\,\mathrm{F}\, p]\!]_3 = ?$
- $[\![w \models \mathrm{F}\,\mathrm{G}\, p]\!]_3 = ?$

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Section

Monitorable Properties

- (Co-)Safety
- Examples
- Monitorability

Monitorability

When Does Anticipation Help?

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

The Good, The Bad and The Ugly

Definition (Good, Bad and Ugly Prefixes)

Given a language $L \subseteq \Sigma^\omega$ of infinite words over Σ we call a finite word $u \in \Sigma^*$

- a good prefix for L if $\forall w \in \Sigma^\omega : uw \in L$,
- a bad prefix for L if $\forall w \in \Sigma^\omega : uw \notin L$ and
- an ugly prefix for L if $\forall v \in \Sigma^* : uv$ is neither a good prefix nor a bad prefix.

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Examples for Good, Bad and Ugly

Examples (The Good, The Bad and The Ugly)

- $\{p\}\{q\}$ is a good prefix for $\mathcal{L}(\mathrm{F}\, q)$.
- $\{p\}\{q\}\{p\}$ is a good prefix for $\mathcal{L}(\mathrm{F}\, q)$.
- $\{p\}\{q\}$ is a bad prefix for $\mathcal{L}(\mathrm{G}\, p)$.
- every $w \in \Sigma^*$ is an ugly prefix for $\mathcal{L}(\mathrm{G}\,\mathrm{F}\, p)$.
- $\{p\}$ is an ugly prefix for $\mathcal{L}(p \rightarrow \mathrm{G}\,\mathrm{F}\, p)$.

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

LTL_3 indentifies good/bad prefixes

Given an LTL formula φ and a finite word $u \in \Sigma^*$, then

$$[\![u \models \varphi]\!]_3 = \begin{cases} \top & \text{if } u \text{ is a good prefix for } \mathcal{L}(\varphi) \\ \bot & \text{if } u \text{ is a bad prefix for } \mathcal{L}(\varphi) \\ ? & \text{otherwise} \end{cases}$$

The Idea of Saftey and Co-Safety

Safety Properties assert that nothing bad happens.

Such a property is violated iff something bad happens after finitely many steps.
($\rightarrow$ A bad prefix exists.)

Co-Safety Properties assert that something good happens.

Such a property is fulfilled iff something good happens after finitely many steps.
($\rightarrow$ A good prefix exists.)

(Co-)Safety Languages and Properties

Definition ((Co-)Safety Languages)

A language $L \subseteq \Sigma^\omega$ is called

- a *safety language* if for all $w \notin L$ there is a prefix $u \in \Sigma^*$ of w which is a bad prefix for L.
- a *co-safety language* if for all $w \in L$ there is a prefix $u \in \Sigma^*$ of w which is a good prefix for L.

Definition ((Co-)Safety Properties)

An LTL formula φ is called

- a *safety property* if its set of models $\mathcal{L}(\varphi)$ is a safety language.
- a *co-safety property* if its set of models $\mathcal{L}(\varphi)$ is a co-safety language.

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$		
$\mathrm{F}\, p$		
$\mathrm{X}\, p$		
$\mathrm{G}\,\mathrm{F}\, p$		
$\mathrm{F}\,\mathrm{G}\, p$		
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$		
$p\,\mathrm{U}\, q$		
$p\,\mathrm{R}\, q$		

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL_3

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$	✔	✘
$\mathrm{F}\, p$		
$\mathrm{X}\, p$		
$\mathrm{G}\,\mathrm{F}\, p$		
$\mathrm{F}\,\mathrm{G}\, p$		
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$		
$p\, \mathrm{U}\, q$		
$p\, \mathrm{R}\, q$		

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$	✔	✘
$\mathrm{F}\, p$	✘	✔
$\mathrm{X}\, p$		
$\mathrm{G}\,\mathrm{F}\, p$		
$\mathrm{F}\,\mathrm{G}\, p$		
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$		
$p\,\mathrm{U}\, q$		
$p\,\mathrm{R}\, q$		

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL$_3$

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$	✔	✘
$\mathrm{F}\, p$	✘	✔
$\mathrm{X}\, p$	✔	✔
$\mathrm{G}\,\mathrm{F}\, p$		
$\mathrm{F}\,\mathrm{G}\, p$		
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$		
$p\, \mathrm{U}\, q$		
$p\, \mathrm{R}\, q$		

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL_3

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$	✔	✘
$\mathrm{F}\, p$	✘	✔
$\mathrm{X}\, p$	✔	✔
$\mathrm{G}\,\mathrm{F}\, p$	✘	✘
$\mathrm{F}\,\mathrm{G}\, p$		
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$		
$p\,\mathrm{U}\, q$		
$p\,\mathrm{R}\, q$		

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$	✔	✘
$\mathrm{F}\, p$	✘	✔
$\mathrm{X}\, p$	✔	✔
$\mathrm{G}\,\mathrm{F}\, p$	✘	✘
$\mathrm{F}\,\mathrm{G}\, p$	✘	✘
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$		
$p\, \mathrm{U}\, q$		
$p\, \mathrm{R}\, q$		

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL_3

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$	✔	✘
$\mathrm{F}\, p$	✘	✔
$\mathrm{X}\, p$	✔	✔
$\mathrm{G}\,\mathrm{F}\, p$	✘	✘
$\mathrm{F}\,\mathrm{G}\, p$	✘	✘
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$	✘	✘
$p\,\mathrm{U}\,q$		
$p\,\mathrm{R}\,q$		

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$	✔	✘
$\mathrm{F}\, p$	✘	✔
$\mathrm{X}\, p$	✔	✔
$\mathrm{G}\,\mathrm{F}\, p$	✘	✘
$\mathrm{F}\,\mathrm{G}\, p$	✘	✘
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$	✘	✘
$p\,\mathrm{U}\,q$	✘	✔
$p\,\mathrm{R}\,q$		

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

Examples

Consider propositions $p, q \in \mathrm{AP}$.

Formula	Safety	Co-Safety
$\mathrm{G}\, p$	✔	✘
$\mathrm{F}\, p$	✘	✔
$\mathrm{X}\, p$	✔	✔
$\mathrm{G}\,\mathrm{F}\, p$	✘	✘
$\mathrm{F}\,\mathrm{G}\, p$	✘	✘
$\mathrm{X}\, p \vee \mathrm{G}\,\mathrm{F}\, p$	✘	✘
$p\,\mathrm{U}\,q$	✘	✔
$p\,\mathrm{R}\,q$	✔	✘

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

Details on The Examples

- $p \, \mathrm{U} \, q$ is not a safety property, because
  $\{p\}^\omega \not\models p \, \mathrm{U} \, q$, but
  there is no bad prefix.
- $p \, \mathrm{U} \, q$ is a co-safety property, because
  every infinite word $w \in \Sigma^\omega$ with $w \models p \, \mathrm{U} \, q$
  must contain the releasing q in a finite prefix.
- $p \, \mathrm{R} \, q$ is not a co-safety property, because
  $\{q\}^\omega \models p \, \mathrm{R} \, q$, but
  there is no good prefix.
- $p \, \mathrm{R} \, q$ is a safety property, because
  every infinite word $w \in \Sigma^\omega$ with $w \not\models p \, \mathrm{R} \, q$
  must contain the violating absence of q in a finite prefix.

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

Monitorability

Definition (Monitorable Languages)

A language $L \subseteq \Sigma^\omega$ is called *monitorable* iff L has no ugly prefix.

Definition (Monitorable Properties)

An LTL formula φ is called *monitorable* iff its set of models $\mathcal{L}(\varphi)$ is monitorable.

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

Monitorable Properties

Safety Properties

Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

Conclusion

Monitorable Properties

Safety Properties

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

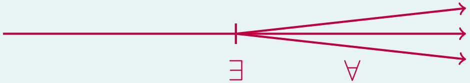
Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

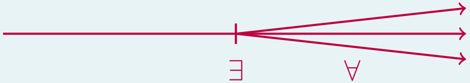
Monitorable Properties

Safety Properties

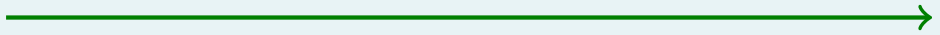


Monitorable Properties

Safety Properties



Co-Safety Properties



Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

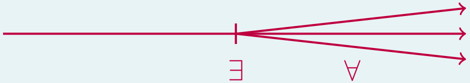
Anticipatory LTL Semantics: LTL$_3$
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

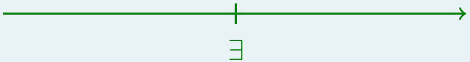
Conclusion

Monitorable Properties

Safety Properties



Co-Safety Properties



Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

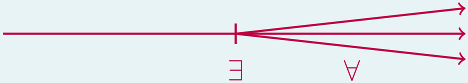
Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

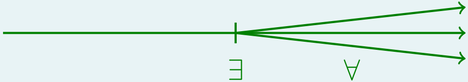
Conclusion

Monitorable Properties

Safety Properties



Co-Safety Properties



Targets & Outline

LTL on Infinite Words
- Semantics
- Equivalences and Examples

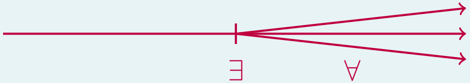
Anticipatory LTL Semantics: LTL_3
- Anticipation
- Definition
- Examples

Monitorable Properties
- (Co-)Safety
- Examples
- Monitorability

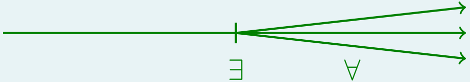
Conclusion

Monitorable Properties

Safety Properties



Co-Safety Properties



Remark

Safety and Co-Safety Properties are monitorable.

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Safety- and Co-Safety-Properties

Theorem

The class of monitorable properties

- *comprises safety- and co-safety properties, but*
- *is strictly larger than their union.*

Targets & Outline

LTL on Infinite Words

Semantics

Equivalences and Examples

Anticipatory LTL Semantics: LTL_3

Anticipation

Definition

Examples

Monitorable Properties

(Co-)Safety

Examples

Monitorability

Conclusion

Safety- and Co-Safety-Properties

Proof.

Consider $\mathrm{AP} = \{p, q, r\}$ and $\varphi = ((p \vee q) \,\mathrm{U}\, r) \vee \mathrm{G}\, p$.

- $\{p\}^\omega \models \varphi$ without good prefix, therefore φ is not a co-safety property.
- $\{q\}^\omega \not\models \varphi$ without bad prefix, therefore φ is not a safety property.
- Every finite word $u \in \Sigma^*$ that is not a bad prefix can become a good prefix by appending $\{r\}$.
- Every finite word $u \in \Sigma^*$ that is not a good prefix can become a bad prefix by appending $\emptyset$.
- No ugly prefix exists as every prefix is either good, bad or can become good or bad by appending $\{r\}$ or $\emptyset$.

□

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Safety- and Co-Safety-Properties

Proof by Another Counterexample.

Consider $\mathrm{AP} = \{p, q\}$ and $\varphi = \mathrm{F}\, p \vee \mathrm{G}\, q$.

- $\{q\}^\omega \models \varphi$ without good prefix,
  therefore φ is not a co-safety property.
- $\emptyset^\omega \not\models \varphi$ without bad prefix,
  therefore φ is not a safety property.
- Every finite word $u \in \Sigma^*$ that is not a bad prefix
  can become a good prefix by appending $\{p\}$.
- No ugly prefix exists as every prefix
  is either bad or can become good by appending $\{p\}$.

□

Targets & Outline

LTL on Infinite Words
Semantics
Equivalences and Examples

Anticipatory LTL Semantics: LTL_3
Anticipation
Definition
Examples

Monitorable Properties
(Co-)Safety
Examples
Monitorability

Conclusion

Conclusion

1. In the semantics for LTL on infinite words there is no difference between X and $\overline{\mathrm{X}}$.
2. The semantics LTL_3 for finite, non-terminated words is defined based on the LTL semantics for infinite words on the lattice $\mathbb{B}_3 = \{\top, ?, \bot\}$.
3. FLTL_4 is not anticipatory, LTL_3 is.
4. $[\![w \models \varphi]\!]_3$ is $\top$ for all good prefixes of $\mathcal{L}(\varphi)$, $\bot$ for all bad prefixes of $\mathcal{L}(\varphi)$ and $?$ for all ugly prefixes of $\mathcal{L}(\varphi)$.
5. The class of monitorable properties comprises safety- and co-safety properties, but is strictly larger than their union.

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Chapter 6

Alternating Büchi Automata

Course “Runtime Verification”

M. Leucker & V. Stolz

INF5140 / V17

Chapter 6

Learning Targets of Chapter "Alternating Büchi Automata".

1. Recall the definition of DFA and NFA.
2. Understand the acceptance condition of Büchi automata (BA) on infinite words.
3. Learn about alternating Büchi automata (ABA).
4. Know how ABA can be translated into BA.

Chapter 6

Outline of Chapter "Alternating Büchi Automata".

Automata on Finite Words

- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

- The Idea
- Definition
- Translating ABA into BA

Section

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Chapter 6 “Alternating Büchi Automata”

Course “Runtime Verification”

M. Leucker & V. Stolz

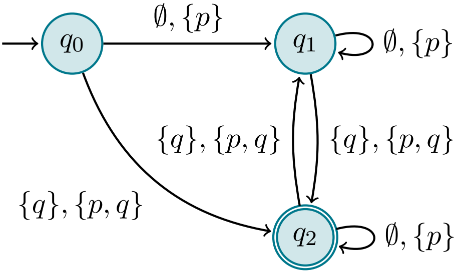
INF5140 / V17

Recall Finite Automata

Finite Automata

- consist of finitely many states and transitions.
- read a word over an alphabet letter by letter.
- change current state depending on the current letter.
- accept or reject a word depending after reading it.

Deterministic Finite Automata



Targets & Outline

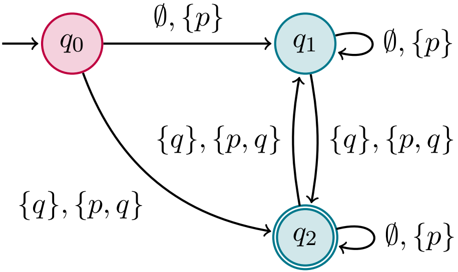
Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Deterministic Finite Automata



Word

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

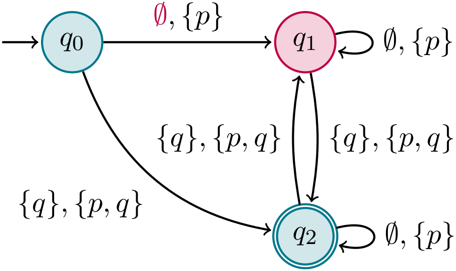
The Idea

Definition

Translating ABA into BA

Conclusion

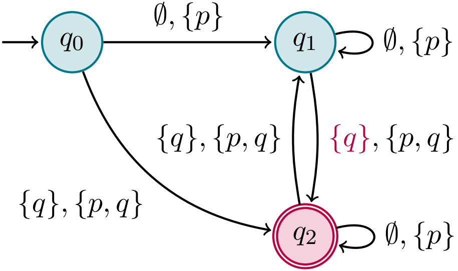
Deterministic Finite Automata



Word

$\emptyset$

Deterministic Finite Automata



Word

$\emptyset\{q\}$

Targets & Outline

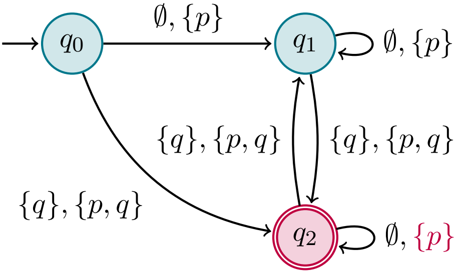
Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Deterministic Finite Automata



Word

$\emptyset\{q\}\{p\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

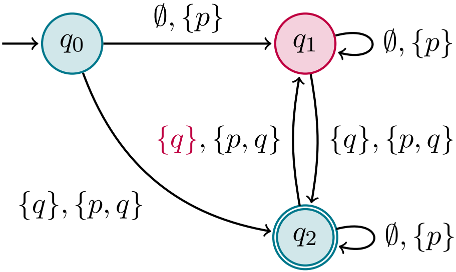
The Idea

Definition

Translating ABA into BA

Conclusion

Deterministic Finite Automata



Word

$\emptyset\{q\}\{p\}\{q\}$

Runtime Verification

M. Leucker & V. Stolz

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

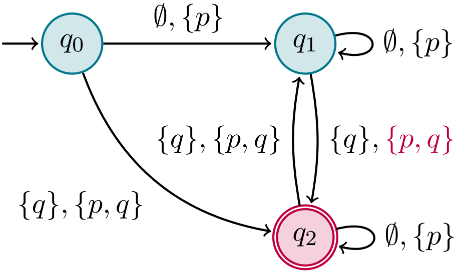
The Idea

Definition

Translating ABA into BA

Conclusion

Deterministic Finite Automata



Word

$\emptyset\{q\}\{p\}\{q\}\{p, q\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Deterministic Finite Automata (DFA)

Definition (Deterministic Finite Automata (DFA))

A *deterministic finite automata (DFA)* is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$ such that

- Σ is the *input alphabet*,
- Q is a finite non-empty set of *states*,
- $q_0 \in Q$ is the *initial state*,
- $\delta : Q \times \Sigma \to Q$ is the *transition function* and
- $F \subseteq Q$ is the set of *accepting states*.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Run of a DFA

Definition (Run of a DFA)

A *Run* of a DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$ on a finite word $w \in \Sigma^*$ is a function $\rho : \{0, \ldots, |w|\} \to Q$ such that

- $\rho(0) = q_0$ and
- $\forall i \in \{1, \ldots, |w|\} : \rho(i) = \delta(\rho(i-1), w_i)$.

A run is called *accepting* iff $\rho(|w|) \in F$.
$\mathcal{A}$ accepts w if the run ρ of $\mathcal{A}$ on w is accepting.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

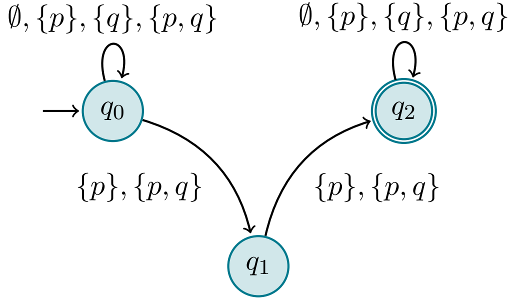
The Idea

Definition

Translating ABA into BA

Conclusion

Non-determinstic Finite Automata



Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

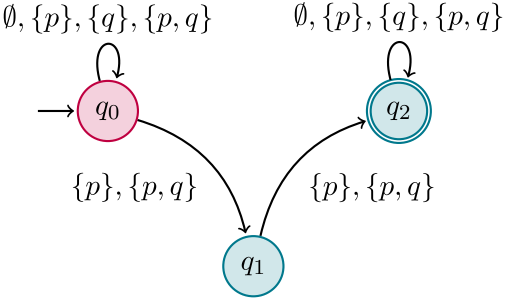
The Idea

Definition

Translating ABA into BA

Conclusion

Non-determinstic Finite Automata



Word

Targets & Outline

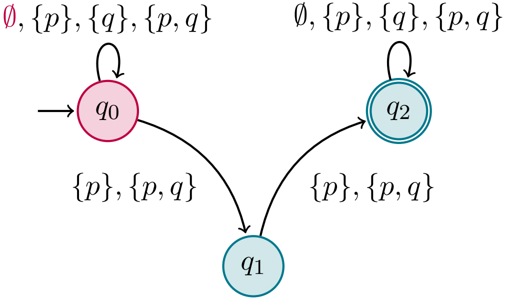
Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Non-determinstic Finite Automata



Word

$\emptyset$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

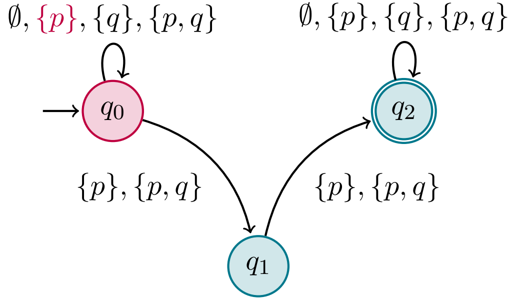
The Idea

Definition

Translating ABA into BA

Conclusion

Non-determinstic Finite Automata



Word

$\emptyset\{p\}$

Targets & Outline

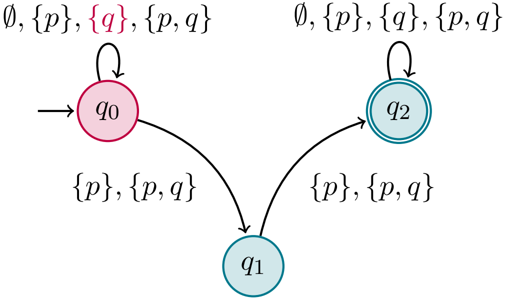
Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Non-determinstic Finite Automata



Word

$\emptyset\{p\}\{q\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

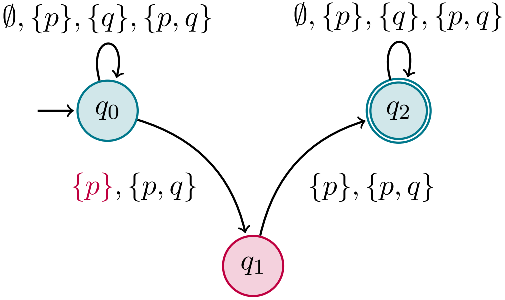
The Idea

Definition

Translating ABA into BA

Conclusion

Non-determinstic Finite Automata



Word

$\emptyset\{p\}\{q\}\{p\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

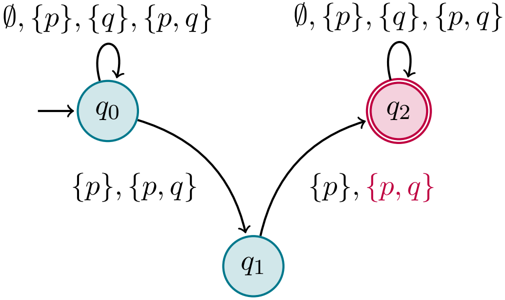
The Idea

Definition

Translating ABA into BA

Conclusion

Non-determinstic Finite Automata



Word

$\emptyset\{p\}\{q\}\{p\}\{p,q\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Non-deterministic Finite Automata (NFA)

Definition (Non-deterministic Finite Automata (NFA))

A *non-deterministic finite automata (NFA)* is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$ such that

- Σ is the *input alphabet*,
- Q is a finite non-empty set of *states*,
- $Q_0 \subseteq Q$ is the set of *initial states*,
- $\Delta \subseteq Q \times \Sigma \times Q$ is the *transition relation* and
- $F \subseteq Q$ is the set of *accepting states*.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Run of an NFA

Definition (Run of an NFA)

A *Run* of an NFA $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$ on a finite word $w \in \Sigma^*$ is a function $\rho : \{0, \ldots, |w|\} \to Q$ such that

- $\rho(0) \in Q_0$ and
- $\forall i \in \{1, \ldots, |w|\} : (\rho(i-1), w_i, \rho(i)) \in \Delta$.

A run is called *accepting* iff $\rho(|w|) \in F$.
$\mathcal{A}$ accepts w if there is an accepting run ρ of $\mathcal{A}$ on w.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Language of Automata

Definition (Language of Automata)

The *language* $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ of an automaton $\mathcal{A}$ with the alphabet Σ is defined as follows:

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ accepts } w\}.$$

We say that $\mathcal{A}$ accepts a language L iff $\mathcal{L}(\mathcal{A}) = L$.

Targets & Outline

Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Power of Finite Automata

- Every NFA can be translated into an equivalent DFA using the power set construction.
- DFAs can accept all regular languages.
  (Proof: Construct a DFA from a regular grammar using its nonterminals as states.)
- For every regular language one can construct a DFA.
  (Proof: See regular expressions.)

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Section

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)
Deterministic Büchi Automata?

Chapter 6 “Alternating Büchi Automata”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

ω-regular Languages

Definition (ω-regular Languages)

A language $L \subseteq \Sigma^\omega$ over an alphabet Σ is called *ω-regular* iff there are regular languages $U_i, V_i \subseteq \Sigma^*$ for $i \in \{1, \dots, m\}$ such that

$$L = \bigcup_{i=1}^{m} U_i \circ V_i^\omega.$$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

ω-regular Languages

Definition (ω-regular Languages)

A language $L \subseteq \Sigma^\omega$ over an alphabet Σ is called *ω-regular* iff there are regular languages $U_i, V_i \subseteq \Sigma^*$ for $i \in \{1, \ldots, m\}$ such that

$$L = \bigcup_{i=1}^{m} U_i \circ V_i^\omega.$$

Example (ω-regular Languages)

Consider an alphabet $\Sigma = 2^{\mathrm{AP}}$ for $\mathrm{AP} = \{p, q\}$.

$$\mathcal{L}(\mathrm{G}\, p) = \{\{p\}, \{p, q\}\}^\omega$$
$$\mathcal{L}(\mathrm{F}\, p) = \Sigma^* \circ \{\{p\}, \{p, q\}\} \circ \Sigma^\omega$$

Finite Automata on Infinite Words

- Finite automata can be used on infinite words as well.
- Definition of run ρ can be reused.
- Problem: There is no last state $\rho(|w|)$.
- Solution: New acceptance condition.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Acceptance Condition of Büchi Automata

- Büchi automata can accept all ω-regular languages.
- An accepting run of a Büchi automaton visits at least one accepting state infinitely often.

ω-regular Languages

- union $\bigcup$
- concatenation $\circ$
- regular language U_i
- regular language V_i^ω

Büchi Automata

- non-determinism
- simple transition
- like an NFA
- new acceptance condition

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

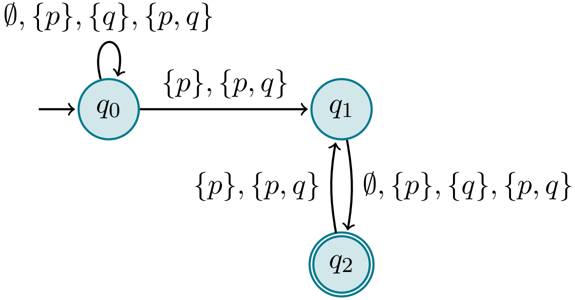
The Idea

Definition

Translating ABA into BA

Conclusion

Non-deterministic Büchi Automata



Targets & Outline

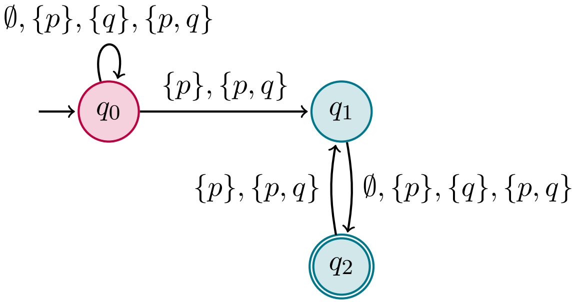
Automata on Finite Words
Deterministic Finite Automata (DFA)
Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
Non-deterministic Büchi Automata (BA)
Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
The Idea
Definition
Translating ABA into BA

Conclusion

Non-deterministic Büchi Automata



Word

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

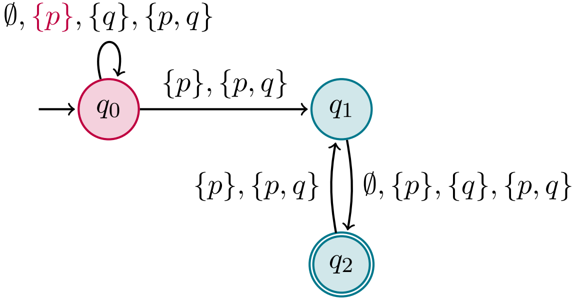
The Idea

Definition

Translating ABA into BA

Conclusion

Non-deterministic Büchi Automata



Word

$\{p\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

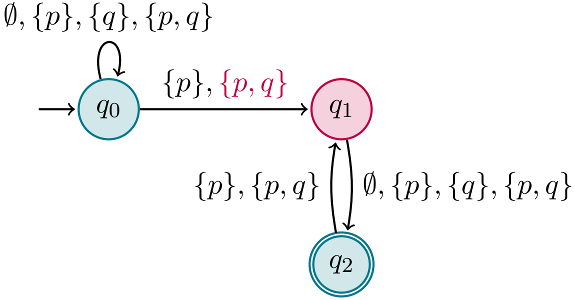
The Idea

Definition

Translating ABA into BA

Conclusion

Non-deterministic Büchi Automata



Word

$\{p\}\{p,q\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

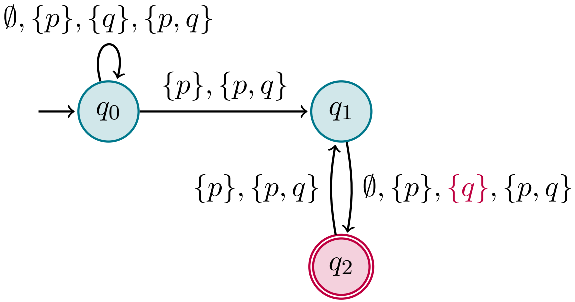
The Idea

Definition

Translating ABA into BA

Conclusion

Non-deterministic Büchi Automata



Word

$\{p\}\{p,q\}\ \{q\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

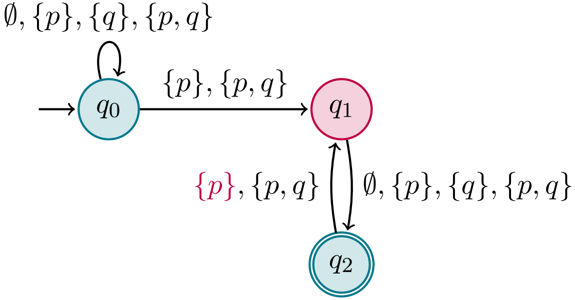
The Idea

Definition

Translating ABA into BA

Conclusion

Non-deterministic Büchi Automata



Word

$\{p\}\{p, q\}\ \{q\}\{p\}$

Targets & Outline

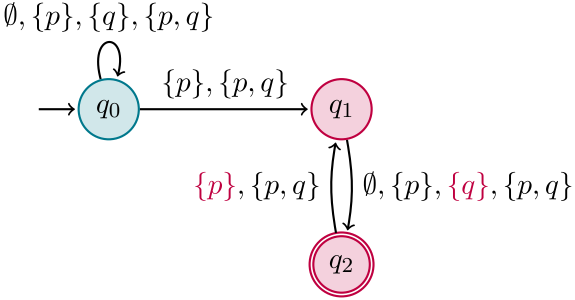
Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Non-deterministic Büchi Automata



Word

$\{p\}\{p,q\}(\{q\}\{p\})^\omega$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Non-deterministic Büchi Automata (BA)

Definition (Non-deterministic Büchi Automata (BA))

A (non-deterministic) *Büchi automaton* is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$ such that

- Σ is the *input alphabet*,
- Q is the finite non-empty set of *states*,
- $Q_0 \subseteq Q$ is the set of *initial states*,
- $\Delta \subseteq Q \times \Sigma \times Q$ is the *transition relation* and
- $F \subseteq Q$ is the set of *accepting states*.

Run of a BA

Definition (Run of a BA)

A run of a BA $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$ on an infinite word $w \in \Sigma^\omega$ is a function $\rho : \mathbb{N} \to Q$ such that

- $\rho(0) \in Q_0$ and
- $\forall i \in \mathbb{N} \backslash \{0\} : (\rho(i-1), w_i, \rho(i)) \in \Delta$.

Targets & Outline

Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Accepting Runs of a BA

Definition (Accepting Runs of a BA)

A run ρ of a BA $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$ is called *accepting* iff

$$\mathrm{Inf}(\rho) \cap F \neq \emptyset,$$

where $\mathrm{Inf}(\rho)$ denotes the *set of states visited infinitely often* given by

$$\mathrm{Inf}(\rho) = \left\{ q \in Q \;\middle|\; |\{k \in \mathbb{N} \mid \rho(k) = q\}| = \infty \right\}.$$

$\mathcal{A}$ accepts w if there is an accepting run ρ of $\mathcal{A}$ on w.

Accepting Runs of a BA

Definition (Accepting Runs of a BA)

A run ρ of a BA $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$ is called *accepting* iff

$$\mathrm{Inf}(\rho) \cap F \neq \emptyset,$$

where $\mathrm{Inf}(\rho)$ denotes the *set of states visited infinitely often* given by

$$\mathrm{Inf}(\rho) = \Big\{ q \in Q \;\Big|\; |\{k \in \mathbb{N} \mid \rho(k) = q\}| = \infty \Big\}.$$

$\mathcal{A}$ accepts w if there is an accepting run ρ of $\mathcal{A}$ on w.

Again the language $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ of an automata $\mathcal{A}$ with the alphabet Σ is defined as follows:

$$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } w \}.$$

Deterministic Büchi Automata

Definition (Deterministic Büchi Automata)

A BA $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$ is called *determistic* iff for every $q \in Q$ and $a \in \Sigma$ there is exactly one $q' \in Q$ such that $(q, a, q') \in \Delta$.

- The successor state is determined by the current state and the action.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Deterministic Büchi Automata

Definition (Deterministic Büchi Automata)

A BA $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$ is called *determistic* iff for every $q \in Q$ and $a \in \Sigma$ there is exactly one $q' \in Q$ such that $(q, a, q') \in \Delta$.

- The successor state is determined by the current state and the action.

- Non-deterministic BA are strictly more expressive than deterministic BA.
- The language $\{p, q\}^* \circ \{q\}^\omega$ cannot be accepted by a deterministic BA.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

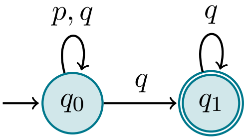
The Idea

Definition

Translating ABA into BA

Conclusion

BA for $\{p,q\}^* \circ \{q\}^\omega$



- Impossible to handle with determistic transition function.
- There are infinitely many possible finite prefixes in $\{p,q\}^*$.
- You don't know when to stop scanning the finite prefix and start scanning the infinite loop.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Section

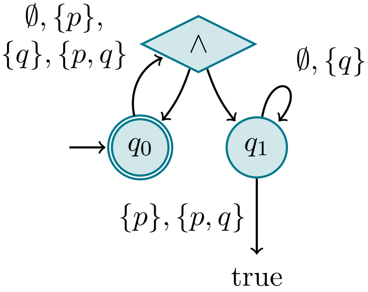
Alternating Büchi Automata (ABA)

The Idea
Definition
Translating ABA into BA

Alternating Automata

- Extend non-deterministic automata by universal choices.
- Non-deterministic transition relations denote a set of possible next states.
- Alternating transition functions denote a positive Boolean combination of next states.

Alternating Büchi Automaton



Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

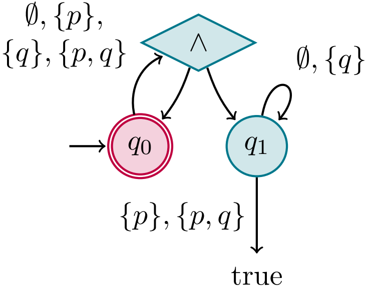
The Idea

Definition

Translating ABA into BA

Conclusion

Alternating Büchi Automaton



Word

Targets & Outline

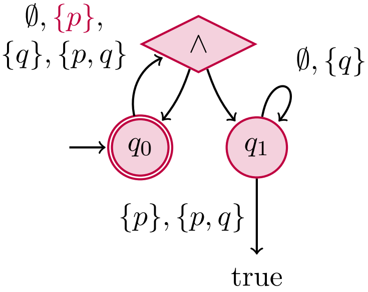
Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Alternating Büchi Automaton



Word

$\{p\}$

Targets & Outline

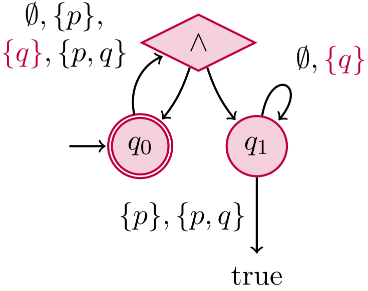
Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

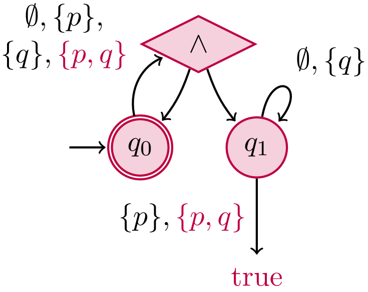
Alternating Büchi Automaton



Word

$\{p\}\{q\}$

Alternating Büchi Automaton



Word

$\{p\}\{q\}\{p, q\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

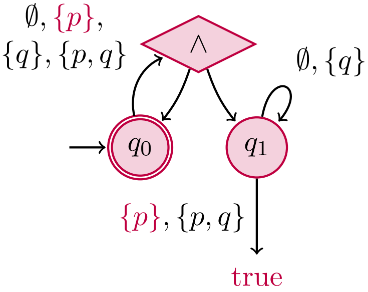
The Idea

Definition

Translating ABA into BA

Conclusion

Alternating Büchi Automaton



Word

$\{p\}\{q\}\{p,q\}\{p\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

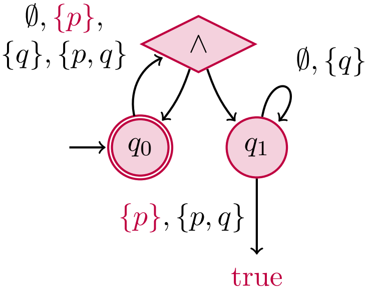
The Idea

Definition

Translating ABA into BA

Conclusion

Alternating Büchi Automaton



Word

$\{p\}\{q\}\{p,q\}\{p\}^\omega$

Runtime Verification

M. Leucker & V. Stolz

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Recall: Positive Boolean Combination

The set $B^+(Q)$ of all positive boolean combinations (PBC) over Q contains

- true, false and all elements of Q and
- all conjunctions and disjunctions of its elements.

A subset $S \subseteq Q$ is a model of $\alpha \in B^+(Q)$, denoted by $S \models \alpha$, iff α evaluates to true in propositional logic interpreting all $p \in S$ as true and all $p \in Q \backslash S$ as false.

Targets & Outline

Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Alternating Büchi Automata (ABA)

Definition (Alternating Büchi Automata (ABA))

A *alternating Büchi automaton* is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ such that

- Σ is the *input alphabet*,
- Q is the finite non-empty set of *states*,
- $Q_0 \in B^+(Q)$ is the PBC of *initial states*,
- $\delta : Q \times \Sigma \to B^+(Q)$ is the *transition function* and
- $F \subseteq Q$ is the set of *accepting states*.

Targets & Outline

Automata on Finite Words
Deterministic Finite Automata (DFA)
Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
Non-deterministic Büchi Automata (BA)
Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
The Idea
Definition
Translating ABA into BA

Conclusion

Minimal Models of a PBC

Definition (Minimal Models of a PBC)

A model $S \models \alpha$ of a PBC $\alpha \in B^+(Q)$ is called minimal, denoted by $S \models\!\!\equiv \alpha$, if none of its proper subsets $S' \subsetneq S$ is a model $S' \models \alpha$.

Examples

Consider $\alpha = (q_1 \wedge q_2) \vee (q_1 \wedge q_3) \in B^+(\{q_0, q_1, q_2, q_3\})$

- $\{q_1, q_2\} \models\!\!\equiv \alpha$
- $\{q_1, q_3\} \models\!\!\equiv \alpha$
- $\{q_1, q_2, q_3\} \models \alpha$ but $\{q_1, q_2, q_3\} \not\models\!\!\equiv \alpha$

Targets & Outline

Automata on Finite Words
Deterministic Finite Automata (DFA)
Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
Non-deterministic Büchi Automata (BA)
Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
The Idea
Definition
Translating ABA into BA

Conclusion

Minimal Models of a PBC

Definition (Minimal Models of a PBC)

A model $S \models \alpha$ of a PBC $\alpha \in B^+(Q)$ is called minimal, denoted by $S \models\!\!\equiv \alpha$, if none of its proper subsets $S' \subsetneq S$ is a model $S' \models \alpha$.

Examples

- $\emptyset \models\!\!\equiv \text{true}$
- $\forall S \subseteq Q : S \neq \emptyset \Rightarrow S \not\models\!\!\equiv \text{true}$
- $\forall S \subseteq Q : S \not\models\!\!\equiv \text{false}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Run of an ABA

Definition (Run of an ABA)

A run of an ABA $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ on an infinite word $w \in \Sigma^\omega$ is a Q-labeled directed acyclic graph (V, E) such that there exist labelings $l : V \to Q$ and $h : V \to \mathbb{N}$ which satisfy the following properties:

- $\{l(v) \mid v \in h^{-1}(0)\} \models Q_0$.
- $E \subseteq \bigcup_{i \in \mathbb{N}} (h^{-1}(i) \times h^{-1}(i+1))$.
- $\forall v' \in V : h(v') \geq 1 \Rightarrow \{v \in V \mid (v, v') \in E\} \neq \emptyset$.
- $\forall v, v' \in V : v \neq v' \wedge l(v) = l(v') \Rightarrow h(v) \neq h(v')$.
- $\forall v \in V : \{l(v') \mid (v, v') \in E\} \models \delta(l(v), w_{h(v)+1})$.

Targets & Outline

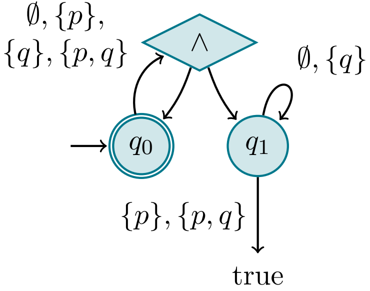
Automata on Finite Words
- Deterministic Finite Automata (DFA)
- Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
- Non-deterministic Büchi Automata (BA)
- Deterministic Büchi Automata?

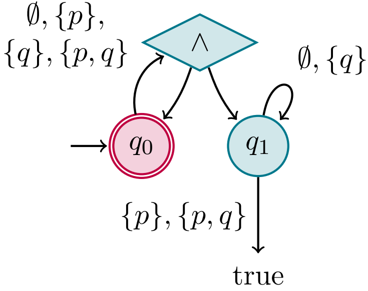
Alternating Büchi Automata (ABA)
- The Idea
- Definition
- Translating ABA into BA

Conclusion

Run of an ABA



Run of an ABA



Word

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

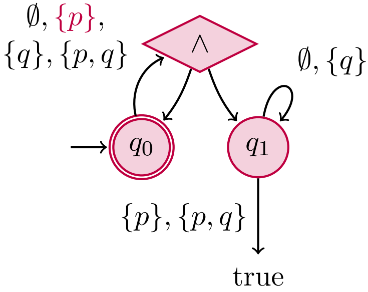
The Idea

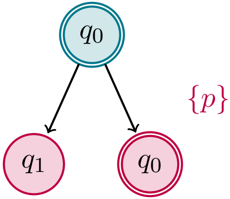
Definition

Translating ABA into BA

Conclusion

Run of an ABA





Word

$\{p\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

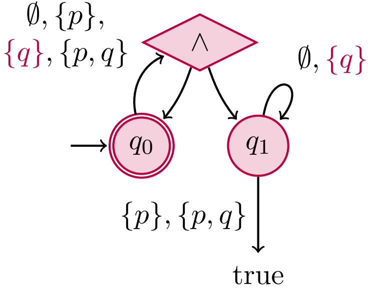
The Idea

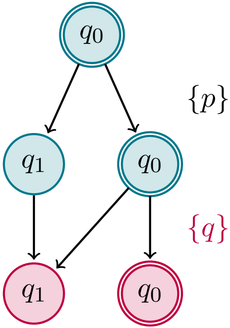
Definition

Translating ABA into BA

Conclusion

Run of an ABA





Word

$\{p\}\{q\}$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

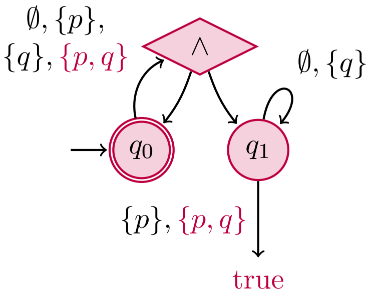
The Idea

Definition

Translating ABA into BA

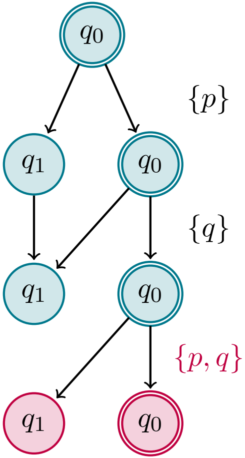
Conclusion

Run of an ABA

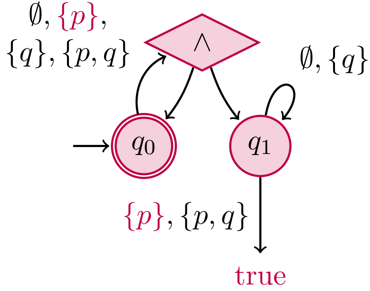


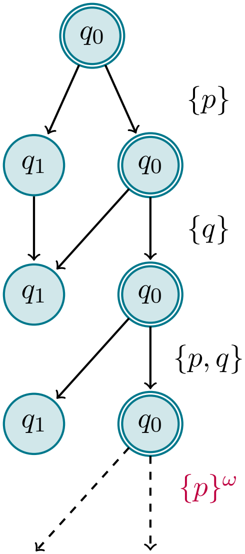
Word

$\{p\}\{q\}\{p, q\}$



Run of an ABA





Word

$\{p\}\{q\}\{p,q\}\{p\}^\omega$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

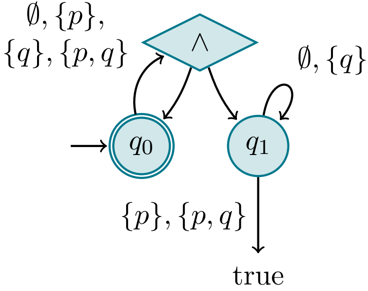
The Idea

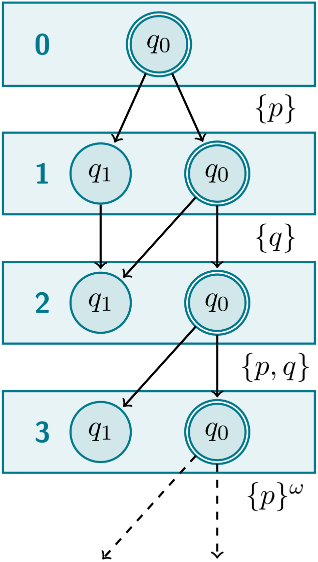
Definition

Translating ABA into BA

Conclusion

Run of an ABA





Word

$\{p\}\{q\}\{p, q\}\{p\}^\omega$

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Accepting Run of an ABA

- A run of a BA is accepting, if infinitely many accepting states are visited.
- Think of the run of an ABA as runs of several copies of BAs at the same time.
- Every copy has to visit infinitely many accepting states.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Accepting Run of an ABA

- A run of a BA is accepting, if infinitely many accepting states are visited.
- Think of the run of an ABA as runs of several copies of BAs at the same time.
- Every copy has to visit infinitely many accepting states.

Definition (Accepting Run of an ABA)

A run (V, E) on $w \in \Sigma^\omega$ is *accepting* if every maximal infinite path, with respect to the labeling l, visits at least one accepting state infinitely often.

Note that every maximal finite path ends in a node $v \in V$ with $\delta(l(v), w_{h(v)+1}) = \text{true}$.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Translating ABA into BA

Theorem

For every alternating Büchi automaton $\mathcal{A}$, there is a Büchi automaton $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. The size of $\mathcal{A}'$ is exponential in the size of $\mathcal{A}$.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

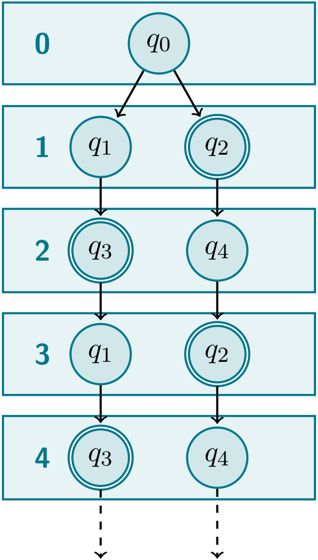
Translating ABA into BA

Conclusion

The Idea

- Transition function of ABA points to PBCs.
- Use minimal models $X \subseteq Q$ of these PBCs as states of the BA.

- Problem: What are the new accepting states?
- Accepting only if every state of X is accepting is not enough.



The Idea: The new accepting states

- Idea: Keep track of the paths on which we already visited an accepting state.
- Split states into two components.
- Shift successor states of the current states into the second component if they are accepting.
- An empty first component indicates that accepting states have been seen on all paths.
- Such states are accepting and all not accpeting successors are shifted back to the first component.

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

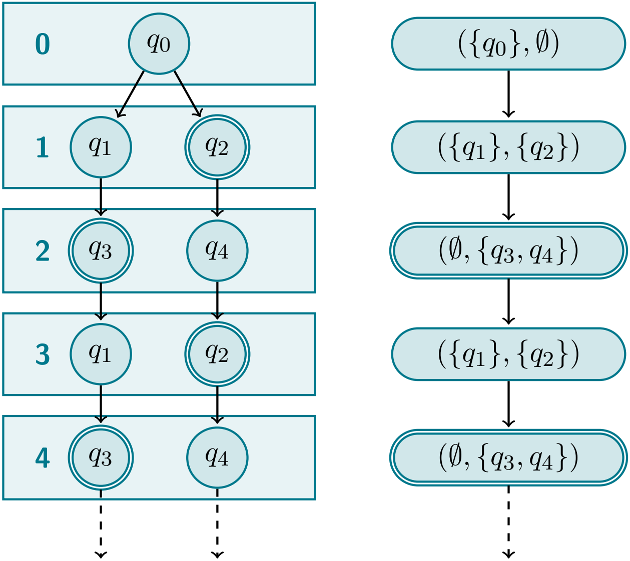
The Idea

Definition

Translating ABA into BA

Conclusion

The Idea: The New Accepting States



Targets & Outline

Automata on Finite Words
Deterministic Finite Automata (DFA)
Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)
Non-deterministic Büchi Automata (BA)
Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)
The Idea
Definition
Translating ABA into BA

Conclusion

Translating ABA into BA

Proof.

Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ be an ABA. We then define the BA $\mathcal{A}' = (\Sigma, Q', Q'_0, \Delta', F')$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$, where

- $Q' = 2^Q \times 2^Q$,
- $\Delta' \subseteq Q' \times \Sigma \times Q'$ as defined next,
- $Q'_0 = \{(X, \emptyset) \mid X \subseteq Q, X \models Q_0\}$ and
- $F' = \emptyset \times 2^Q$.

□

Translating ABA into BA

Proof.

A pair $(U, V) \in Q'$, an action $a \in \Sigma$ and a pair $(U', V') \in Q'$ are element of the transition relation Δ' iff

- case $U \neq \emptyset$: there are $X, Y \subseteq Q$ satisfying

$$X \models \bigwedge_{q \in U} \delta(q, a) \text{ and } Y \models \bigwedge_{q \in V} \delta(q, a)$$

  and $U' = X \setminus F$ and $V' = (X \cap F) \cup (Y \setminus U')$.

- case $U = \emptyset$: there is $Y \subseteq Q$ satisfying

$$Y \models \bigwedge_{q \in V} \delta(q, a)$$

  and $U' = Y \setminus F$ and $V' = Y \cap F$.

Note that we identify an empty conjunction with true, so that e. g. $((\emptyset, \emptyset), a, (\emptyset, \emptyset)) \in \Delta'$.

□

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

The Idea

Definition

Translating ABA into BA

Conclusion

Translating ABA into BA

Proof.

By $Q' = 2^Q \times 2^Q$ we get

$$|Q'| = 2^{|Q|} \cdot 2^{|Q|} = 2^{2|Q|},$$

but since the components U and V of a state (U, V) have an empty intersection $U \cap V = \emptyset$ we get an upper bound for the number of needed states of

$$\sum_{k=0}^{|Q|} \binom{|Q|}{k} \left(\sum_{l=0}^{k} \binom{k}{l} \right)$$
$$= \sum_{k=0}^{|Q|} \binom{|Q|}{k} 2^k$$
$$= 3^{|Q|} = 2^{|Q| \log_2 3}$$

□

Targets & Outline

Automata on Finite Words

Deterministic Finite Automata (DFA)

Non-determinstic Finite Automata (NFA)

Büchi Automata (BA)

Non-deterministic Büchi Automata (BA)

Deterministic Büchi Automata?

Alternating Büchi Automata (ABA)

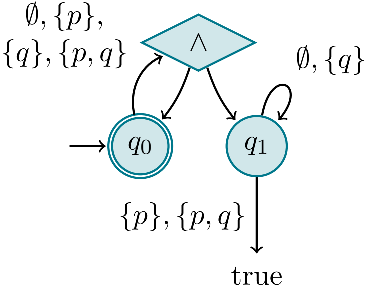
The Idea

Definition

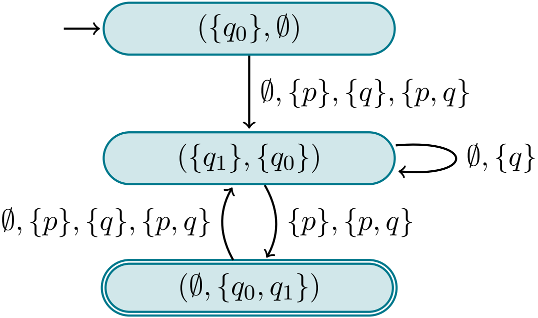
Translating ABA into BA

Conclusion

Example: ABA



Example: Matching BA



Conclusion

1. DFA and NFA are finite automata on finite words and accept a word depending on the last state of a run on it.
2. Büchi automata (BA) are finite automata on infintite words and accept a word depending on the set of states visited infinitely often in a run on it.
3. Non-deterministic BA can accept all ω-regular languages, deterministic BA cannot.
4. Alternating Büchi Automata (ABA) use transition functions mapping to positive Boolean combinations (PBCs) of states.
5. Every ABA can be translated into an BA accepting the same language using minimal models of the PBCs as states and keeping track of the paths on which accepting states were already visited.

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Chapter 7

Monitor Construction for Anticipatory Runtime Verification

Course "Runtime Verification"

M. Leucker & V. Stolz

INF5140 / V17

Chapter 7

Learning Targets of Chapter "Monitor Construction for Anticipatory Runtime Verification".

1. Understand the translation from LTL to alternating Büchi automata.
2. Understand the monitor construction for LTL_3.
3. Know the single steps of this construction, especially how a satisfiability check is implemented in the emptiness per state function.
4. Learn about its complexity and its relation to monitorable properties.

Chapter 7

Outline of Chapter “Monitor Construction for Anticipatory Runtime Verification”.

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Section

The Idea

Impartial Anticipation
The Construction

Impartial Anticipation

Impartiality

- Go for a final verdict ($\top$ or $\bot$) only if you really know.
- Be a rational being: Stick to your word.
- Every two-valued logic is not impartial.
  We therefore use $\mathbb{B}_3 = \{\bot, ?, \top\}$.

Anticipation

- Go for a final verdict ($\top$ or $\bot$) once you really know.
- Don't be a cunctator: Don't delay the decision.
- Implementing anticipation is dificult—you need to identify all shortest bad resp. good prefixes.
- Consider for example $\mathrm{X}\,\mathrm{X}\,\mathrm{X}\,\mathrm{false}$ or $\mathrm{F}\,\mathrm{false}$.

The Semantics

Let φ be an LTL formula and let $u \in \Sigma^*$ be a finite word. Then the semantics of φ with respect to u is given as follows:

$$[\![u \models \varphi]\!]_3 = \begin{cases} \top & \text{if } \forall w \in \Sigma^\omega : [\![uw \models \varphi]\!]_\omega = \top \\ \bot & \text{if } \forall w \in \Sigma^\omega : [\![uw \models \varphi]\!]_\omega = \bot \\ ? & \text{else.} \end{cases}$$

Target

Construct a Moore machine $\mathcal{M}^\varphi$ for an LTL formula φ that

- reads a word letter by letter
- outputs in every state the value of $[\![w \models \varphi]\!]_3$ where w is the word read so far.

First Idea

Remember the evlFLTL_4 function:

- It reads a word letter by letter and
- outputs the subformula that has to be satisfied next for every letter it gets.

First idea: Perform a satisfiability check on the returned formula of such a function.

- Return $?$ if the formula is satisfiable.
- Return $\top$ if the formula is a tautology.
- Return $\perp$ if the formula is a contradiction.

Satisfiability Checking for LTL

- Satisfiability checking for LTL is a difficult task.
- The problem of deciding if there exists a word $w \in \Sigma^\omega$ for an LTL formula φ such that $w \models \varphi$ is PSPACE-complete.
- We will use a translation of LTL formulae to Büchi automata to perform this task.

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

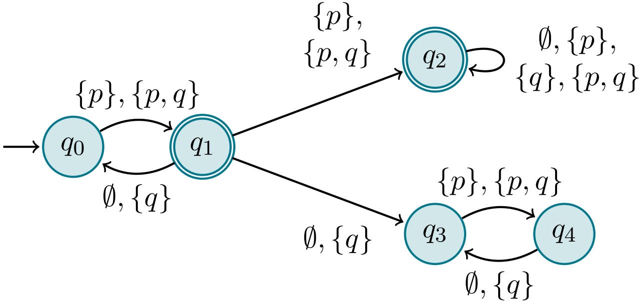
Analysis
Complexity
Monitorable Properties

Conclusion

Monitor construction

Construct a BA of the LTL formula and identify:

For example consider $\mathrm{AP} = \{p, q\}$ and $\Sigma = 2^{\mathrm{AP}}$:



Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

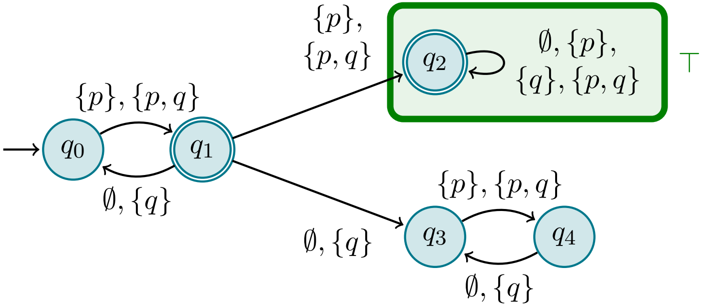
Conclusion

Monitor construction

Construct a BA of the LTL formula and identify:

good states $\top$ BA will accept on every continuation.

For example consider $\mathrm{AP} = \{p, q\}$ and $\Sigma = 2^{\mathrm{AP}}$:



Runtime Verification

M. Leucker & V. Stolz

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

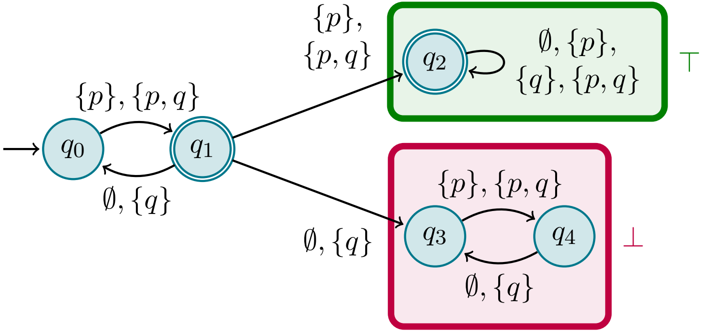
Monitor construction

Construct a BA of the LTL formula and identify:

good states $\top$ BA will accept on every continuation.

bad states $\bot$ BA will reject on every continuation.

For example consider $\mathrm{AP} = \{p, q\}$ and $\Sigma = 2^{\mathrm{AP}}$:



Monitor construction

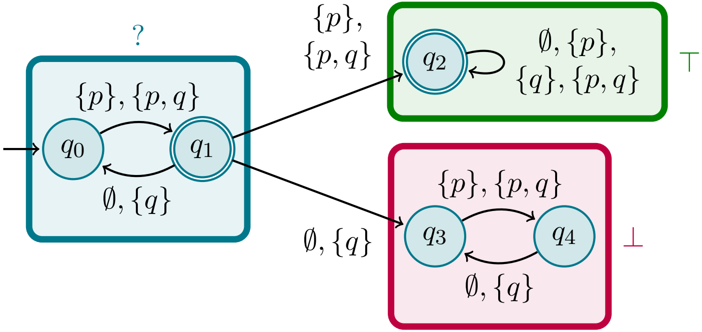
Construct a BA of the LTL formula and identify:

good states $\top$ BA will accept on every continuation.

bad states $\bot$ BA will reject on every continuation.

other states $?$ We don't know (yet).

For example consider $\mathrm{AP} = \{p, q\}$ and $\Sigma = 2^{\mathrm{AP}}$:



Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Monitor construction

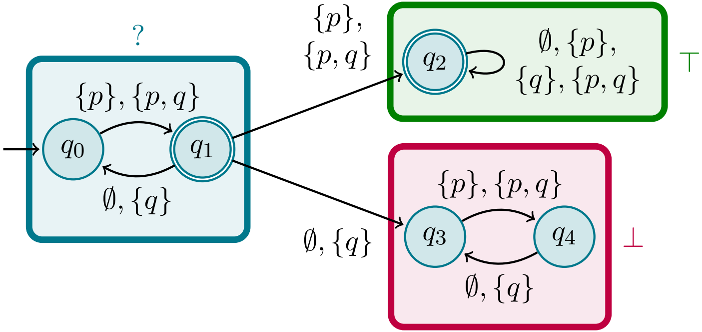
Construct a BA of the LTL formula and identify:

good states $\top$ BA will accept on every continuation.

bad states $\bot$ BA will reject on every continuation.

other states $?$ We don't know (yet).

For example consider $\mathrm{AP} = \{p, q\}$ and $\Sigma = 2^{\mathrm{AP}}$:



Create a Moore machine using these labels as outputs.

Analysis of the Construction

It's an LTL_3 Monitor!

- ✔ One can construct an ABA from LTL.
- ✔ We can translate an ABA into a BA.
- ✔ The monitor performs the desired satisfiability check.

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Analysis of the Construction

It's an LTL_3 Monitor!

- ✔ One can construct an ABA from LTL.
- ✔ We can translate an ABA into a BA.
- ✔ The monitor performs the desired satisfiability check.

How to detect regions?

- ✔ The bad regions ($\bot$) are subautomata S without accepting states and without edges leaving S.
- ✔ They can be identified using linear-time nested depth-first search algorithms.

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Analysis of the Construction

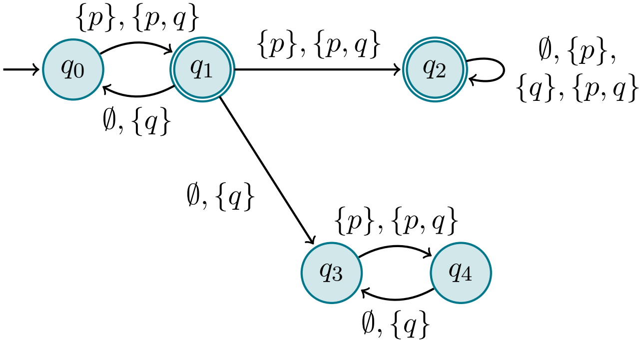
It's an LTL_3 Monitor!

- ✔ One can construct an ABA from LTL.
- ✔ We can translate an ABA into a BA.
- ✔ The monitor performs the desired satisfiability check.

How to detect regions?

- ✔ The bad regions ($\bot$) are subautomata S without accepting states and without edges leaving S.
- ✔ They can be identified using linear-time nested depth-first search algorithms.
- ✘ The good regions ($\top$) are universal subautomata accepting every possible word.
- ✘ Universality check for BA is PSPACE-complete.

Identifying The Good States



Targets & Outline

The Idea
Impartial Anticipation
The Construction

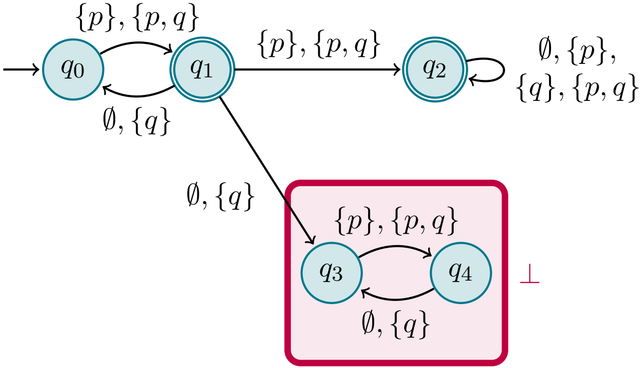
The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

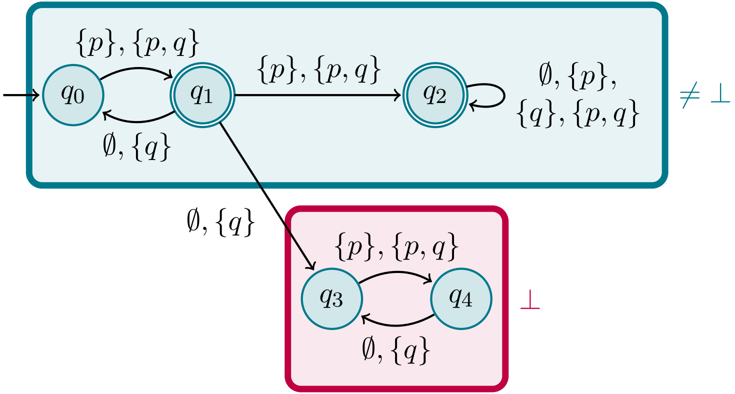
Identifying The Good States

- Only identify bad states (⊥) and



Identifying The Good States

- Only identify bad states ($\bot$) and
- label everything else as not bad ($\neq \bot$).



Targets & Outline

The Idea
Impartial Anticipation
The Construction

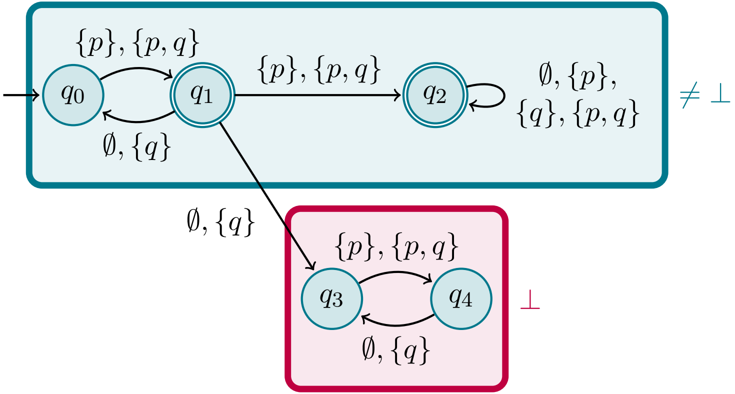
The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Identifying The Good States

- Only identify bad states ($\perp$) and
- label everything else as not bad ($\neq \perp$).
- Perform this for the LTL formulae φ and $\neg\varphi$.
- Good states for φ are bad states for $\neg\varphi$.



Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Identifying The Good States

- Only identify bad states ($\perp$) and
- label everything else as not bad ($\neq \perp$).
- Perform this for the LTL formulae φ and $\neg\varphi$.
- Good states for φ are bad states for $\neg\varphi$.

Remark

- An LTL formula can be complemented by adding $\neg$.
- Computing the NNF can be done in linear time.
- Complementing a BA potentially needs exponential time.

The Complete Construction

1. Translate φ and $\neg\varphi$ into ABA.
2. Translate ABAs into BAs.
3. Create NFAs based on bad states in BAs.
4. Determinize NFAs to DFAs.
5. Compute Moore machine out of both DFAs.

Section

The Construction

From LTL to ABA
Emptiness per State
The Monitor

Recall the Idea of the AMM for FLTL_4

We monitor an LTL formula φ by evaluating its current subformula ψ w.r.t. the current letter a. Progression provides the LTL formula ψ' that has to be fulfilled next.

- The set of states consists of all subformulae of φ.
- The initial state is φ.
- The current state is ψ.
- It reads the letter a
- and sets ψ' as new state.

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Recall the Idea of the AMM for FLTL_4

We monitor an LTL formula φ by evaluating its current subformula ψ w.r.t. the current letter a. Progression provides the LTL formula ψ' that has to be fulfilled next.

- The set of states consists of all subformulae of φ.
- The initial state is φ.
- The current state is ψ.
- It reads the letter a
- and sets ψ' as new state.

But when to accept?

- ABA accepts if all paths are finite (ending in true).
- Only Release R and Globally G allow for infinite loops.
- Make these states accepting.

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

LTL to ABA

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $a \in \Sigma$ a letter, φ, ψ_1, ψ_2 LTL formulae in NNF and Q the set of all subformulae of φ.

We then define $\mathrm{ABA}^{\varphi} = (\Sigma, Q, \varphi, \delta, F)$ with

$$F = \{\psi_1 \,\mathrm{R}\, \psi_2, \mathrm{G}\, \psi_1 \mid \psi_1, \psi_2 \in Q\}.$$

and its transition function $\delta : Q \times \Sigma \to B^{+}(Q)$ inductively given as follows:

$$\delta(\mathrm{true}, a) = \mathrm{true}$$

$$\delta(\mathrm{false}, a) = \mathrm{false}$$

$$\delta(p, a) = \begin{cases} \mathrm{true} & \text{if } p \in a \\ \mathrm{false} & \text{else} \end{cases}$$

$$\delta(\neg p, a) = \begin{cases} \mathrm{true} & \text{if } p \notin a \\ \mathrm{false} & \text{else} \end{cases}$$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

LTL to ABA

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $a \in \Sigma$ a letter, φ, ψ_1, ψ_2 LTL formulae in NNF and Q the set of all subformulae of φ.

We then define $\mathrm{ABA}^{\varphi} = (\Sigma, Q, \varphi, \delta, F)$ with

$$F = \{\psi_1 \,\mathrm{R}\, \psi_2, \mathrm{G}\, \psi_1 \mid \psi_1, \psi_2 \in Q\}.$$

and its transition function $\delta : Q \times \Sigma \to B^{+}(Q)$ inductively given as follows:

$$\delta(\psi_1 \vee \psi_2, a) = \delta(\psi_1, a) \vee \delta(\psi_2, a)$$
$$\delta(\psi_1 \wedge \psi_2, a) = \delta(\psi_1, a) \wedge \delta(\psi_2, a)$$
$$\delta(\mathrm{X}\, \psi_1, a) = \psi_1$$
$$\delta(\overline{\mathrm{X}}\, \psi_1, a) = \psi_1$$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

LTL to ABA

Let $\Sigma = 2^{\mathrm{AP}}$ be the finite alphabet, $p \in \mathrm{AP}$ an atomic proposition, $a \in \Sigma$ a letter, φ, ψ_1, ψ_2 LTL formulae in NNF and Q the set of all subformulae of φ.

We then define $\mathrm{ABA}^{\varphi} = (\Sigma, Q, \varphi, \delta, F)$ with

$$F = \{\psi_1 \,\mathrm{R}\, \psi_2, \mathrm{G}\, \psi_1 \mid \psi_1, \psi_2 \in Q\}.$$

and its transition function $\delta : Q \times \Sigma \to B^+(Q)$ inductively given as follows:

$$\delta(\psi_1 \,\mathrm{U}\, \psi_2, a) = \delta(\psi_2 \vee (\psi_1 \wedge \mathrm{X}(\psi_1 \,\mathrm{U}\, \psi_2)), a)$$
$$\delta(\psi_1 \,\mathrm{R}\, \psi_2, a) = \delta(\psi_2 \wedge (\psi_1 \vee \overline{\mathrm{X}}(\psi_1 \,\mathrm{R}\, \psi_2)), a)$$
$$\delta(\mathrm{F}\, \psi_1, a) = \delta(\psi_1 \vee (\mathrm{X}(\mathrm{F}\, \psi_1)), a)$$
$$\delta(\mathrm{G}\, \psi_1, a) = \delta(\psi_1 \wedge (\overline{\mathrm{X}}(\mathrm{G}\, \psi_1)), a)$$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Example

Consider $\mathrm{AP} = \{p, q\}$, $\Sigma = 2^{\mathrm{AP}}$ and $\varphi = \mathrm{G}(p \,\mathrm{U}\, q)$.

$$\begin{aligned}
\delta(\mathrm{G}(p \,\mathrm{U}\, q), a) &= \mathrm{G}(p \,\mathrm{U}\, q) \wedge \delta(p \,\mathrm{U}\, q, a) \quad \forall a \in \Sigma \\
\delta(p \,\mathrm{U}\, q, \emptyset) &= \delta(q, \emptyset) \vee (\delta(p, \emptyset) \wedge (p \,\mathrm{U}\, q)) \\
&= \mathrm{false} \vee (\mathrm{false} \wedge (p \,\mathrm{U}\, q)) = \mathrm{false} \\
\delta(p \,\mathrm{U}\, q, \{p\}) &= \delta(q, \{p\}) \vee (\delta(p, \{p\}) \wedge (p \,\mathrm{U}\, q)) \\
&= \mathrm{false} \vee (\mathrm{true} \wedge (p \,\mathrm{U}\, q)) = p \,\mathrm{U}\, q \\
\delta(p \,\mathrm{U}\, q, \{q\}) &= \delta(q, \{q\}) \vee (\delta(p, \{q\}) \wedge (p \,\mathrm{U}\, q)) \\
&= \mathrm{true} \vee (\mathrm{false} \wedge (p \,\mathrm{U}\, q)) = \mathrm{true} \\
\delta(p \,\mathrm{U}\, q, \{p, q\}) &= \delta(q, \{p, q\}) \vee (\delta(p, \{p, q\}) \wedge (p \,\mathrm{U}\, q)) \\
&= \mathrm{true} \vee (\mathrm{true} \wedge (p \,\mathrm{U}\, q)) = \mathrm{true}
\end{aligned}$$

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

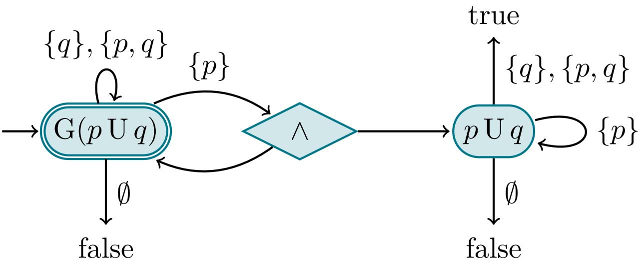
The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Example

Consider $\mathrm{AP} = \{p, q\}$, $\Sigma = 2^{\mathrm{AP}}$ and $\varphi = \mathrm{G}(p \,\mathrm{U}\, q)$.



Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

The Monitor Construction

For a given LTL formula φ over an alphabet Σ we construct a Moore machine $\mathcal{M}^\varphi$ that

- reads finite words $w \in \Sigma^*$ and
- outputs $[\![w \models \varphi]\!]_3 \in \mathcal{B}_3$.

For the next steps let

$$\mathcal{A}^\varphi = (\Sigma, Q^\varphi, Q_0^\varphi, \delta^\varphi, F^\varphi)$$

denote the BA accepting all models of φ and

$$\mathcal{A}^{\neg\varphi} = (\Sigma, Q^{\neg\varphi}, Q_0^{\neg\varphi}, \delta^{\neg\varphi}, F^{\neg\varphi})$$

denote the BA accepting all words falsifying φ.

Targets & Outline
The Idea
Impartial Anticipation
The Construction
The Construction
From LTL to ABA
Emptiness per State
The Monitor
Analysis
Complexity
Monitorable Properties
Conclusion

Emptiness per State

Definition (BA With Adjusted Initial State)

For an BA $\mathcal{A}$, we denote by $\mathcal{A}(q)$ the BA that coincides with $\mathcal{A}$ except for the set of initial states Q_0, which is redefined in $\mathcal{A}(q)$ as $Q_0 = \{q\}$.

Definition (Emptiness per State)

We then define a function $\mathcal{F}^\varphi : Q^\varphi \to \mathbb{B}_2$ (with $\mathbb{B}_2 = \{\top, \bot\}$) where we set $\mathcal{F}^\varphi(q) = \top$ iff $\mathcal{L}(\mathcal{A}^\varphi(q)) \neq \emptyset$.

Emptiness per State

Definition (BA With Adjusted Initial State)

For an BA $\mathcal{A}$, we denote by $\mathcal{A}(q)$ the BA that coincides with $\mathcal{A}$ except for the set of initial states Q_0, which is redefined in $\mathcal{A}(q)$ as $Q_0 = \{q\}$.

Definition (Emptiness per State)

We then define a function $\mathcal{F}^\varphi : Q^\varphi \to \mathbb{B}_2$ (with $\mathbb{B}_2 = \{\top, \bot\}$) where we set $\mathcal{F}^\varphi(q) = \top$ iff $\mathcal{L}(\mathcal{A}^\varphi(q)) \neq \emptyset$.

Using $\mathcal{F}^\varphi$, we define the NFA $\hat{\mathcal{A}}^\varphi = (\Sigma, Q^\varphi, Q_0^\varphi, \delta^\varphi, \hat{F}^\varphi)$ with $\hat{F}^\varphi = \{q \in Q^\varphi \mid \mathcal{F}^\varphi(q) = \top\}$. Analogously, we set $\hat{\mathcal{A}}^{\neg\varphi} = (\Sigma, Q^{\neg\varphi}, Q_0^{\neg\varphi}, \delta^{\neg\varphi}, \hat{F}^{\neg\varphi})$ with $\hat{F}^{\neg\varphi} = \{q \in Q^{\neg\varphi} \mid \mathcal{F}^{\neg\varphi}(q) = \top\}$.

LTL_3 Evaluation

Lemma (LTL_3 Evaluation)

With the notation as before, we have

$$[\![w \models \varphi]\!]_3 = \begin{cases} \top & \text{if } w \notin \mathcal{L}(\hat{\mathcal{A}}^{\neg\varphi}) \\ \bot & \text{if } w \notin \mathcal{L}(\hat{\mathcal{A}}^{\varphi}) \\ ? & \text{if } w \in \mathcal{L}(\hat{\mathcal{A}}^{\varphi} \cap \mathcal{L}(\hat{\mathcal{A}}^{\neg\varphi})) \end{cases}$$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

LTL_3 Evaluation

Proof.

$[\![w \models \varphi]\!]_3 = \top$ if $w \notin \mathcal{L}(\hat{\mathcal{A}}^{\neg\varphi})$

- Feeding a finite prefix $w \in \Sigma^*$ to the BA $\mathcal{A}^{\neg\varphi}$, we reach the set $\delta^{\neg\varphi}(Q_0^{\neg\varphi}, w) \subseteq Q^{\neg\varphi}$ of states.
- If $\exists q \in \delta^{\neg\varphi}(Q_0^{\neg\varphi}, w) : \mathcal{L}(\mathcal{A}^{\neg\varphi}(q)) \neq \emptyset$ then we can choose $\sigma \in \mathcal{L}(\mathcal{A}^{\neg\varphi}(q))$ such that $w\sigma \in \mathcal{L}(\mathcal{A}^{\neg\varphi})$.
- Such a state q exists by definition iff $w \in \mathcal{L}(\hat{\mathcal{A}}^{\neg\varphi})$.
- If $w \notin \mathcal{L}(\hat{\mathcal{A}}^{\neg\varphi})$ then every possible continuation $w\sigma$ of w will be rejected by $\mathcal{A}^{\neg\varphi}$, i.e. $[\![w\sigma \models \varphi]\!]_\omega = \top$ for all $\sigma \in \Sigma^\omega$. Therefore we have $[\![w \models \varphi]\!]_3 = \top$.

$[\![w \models \varphi]\!]_3 = \bot$ if $w \notin \mathcal{L}(\hat{\mathcal{A}}^{\varphi})$

- can be seen by substituting φ for $\neg\varphi$.

□

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

LTL_3 Evaluation

Proof.

$[\![w \models \varphi]\!]_3 = ?$ if $w \in \mathcal{L}(\hat{\mathcal{A}}^{\neg\varphi}) \cap \mathcal{L}(\hat{\mathcal{A}}^{\varphi})$

- If $\exists q \in \delta^{\neg\varphi}(Q_0^{\neg\varphi}, w) : \mathcal{L}(\mathcal{A}^{\neg\varphi}(q)) \neq \emptyset$ and $\exists q' \in \delta^{\varphi}(Q_0^{\varphi}, w) : \mathcal{L}(\mathcal{A}^{\varphi}(q')) \neq \emptyset$ then we can choose $\sigma \in \mathcal{L}(\mathcal{A}^{\neg\varphi}(q))$ and $\sigma' \in \mathcal{L}(\mathcal{A}^{\varphi}(q'))$ such that $[\![w\sigma \models \varphi]\!]_2 = \bot$ and $[\![w\sigma' \models \varphi]\!]_2 = \top$.
- Hence we have $[\![w \models \varphi]\!]_3 = ?$.

□

Targets & Outline

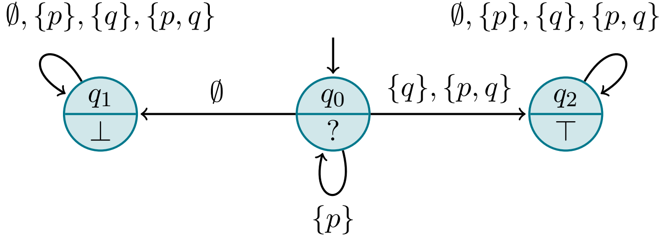
The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

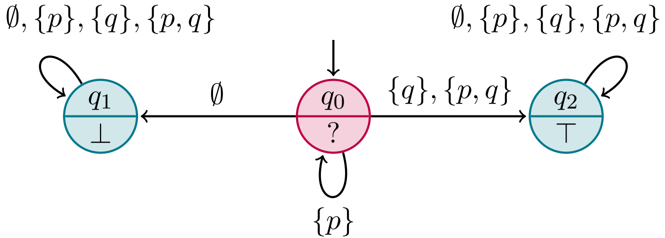
Conclusion

Deterministic Moore Machine (FSM)



Targets & Outline
The Idea
Impartial Anticipation
The Construction
The Construction
From LTL to ABA
Emptiness per State
The Monitor
Analysis
Complexity
Monitorable Properties
Conclusion

Deterministic Moore Machine (FSM)



Input

Output
?

Targets & Outline

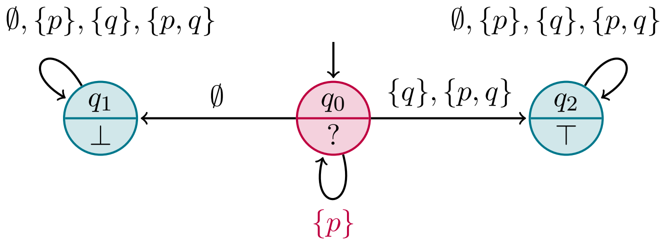
The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

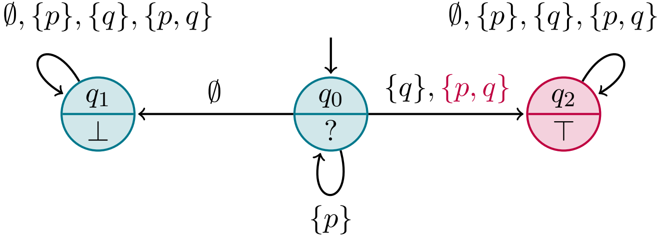
Deterministic Moore Machine (FSM)



Input
$\{p\}$

Output
??

Deterministic Moore Machine (FSM)



Input	Output
$\{p\}\{p, q\}$	$??\top$

Targets & Outline

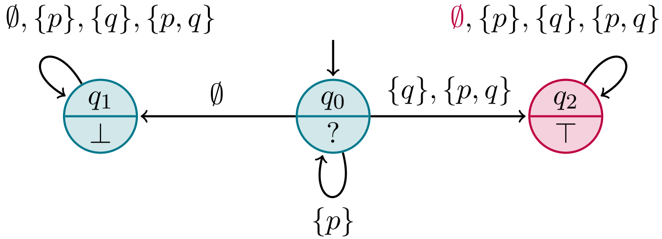
The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Deterministic Moore Machine (FSM)



Input	Output
$\{p\}\{p,q\}\emptyset$	$??\top\top$

Determinstic Moore Machine (FSM)

Definition (Deterministic Moore Machine (FSM))

A (deterministic) *Moore machine* is a tupel $\mathcal{M} = (\Sigma, Q, q_0, \Gamma, \delta, \lambda)$ where

- Σ is the *input alphabet*,
- Q is a finite set of *states*,
- $q_0 \in Q$ is the *initial state*,
- Γ is the *output alphabet*,
- $\delta : Q \times \Sigma \to Q$ is the *transition function* and
- $\lambda : Q \to \Gamma$ is the *output function*.

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Run of a Deterministic Moore Machine

Definition (Run of a Deterministic Moore Machine)

A *run* of a (deterministic) Moore machine $\mathcal{M} = (\Sigma, Q, q_0, \Gamma, \delta, \lambda)$ on a finite word $w \in \Sigma^n$ with outputs $o_i \in \Gamma$ is a sequence

$$t_0 \stackrel{w_1}{\rightarrow} t_1 \stackrel{w_2}{\rightarrow} \ldots \stackrel{w_{n-1}}{\rightarrow} t_{n-1} \stackrel{w_n}{\rightarrow} t_n$$

such that

- $t_0 = q_0$,
- $t_i = \delta(t_{i-1}, w_i)$ and
- $o_i = \lambda(t_i)$.

The *output* of the run is $o_n = \lambda(t_n)$.

Targets & Outline

The Idea

Impartial Anticipation

The Construction

The Construction

From LTL to ABA

Emptiness per State

The Monitor

Analysis

Complexity

Monitorable Properties

Conclusion

Monitor $\mathcal{M}^{\varphi}$ for an LTL Formula φ

Let $\tilde{A}^{\varphi} = (\Sigma, \tilde{Q}^{\varphi}, q_0^{\varphi}, \tilde{\delta}^{\varphi}, \tilde{F}^{\varphi})$ and $\tilde{A}^{\neg\varphi} = (\Sigma, \tilde{Q}^{\neg\varphi}, q_0^{\neg\varphi}, \tilde{\delta}^{\neg\varphi}, \tilde{F}^{\neg\varphi})$ be the equivalent DFAs of the NFAs $\hat{A}^{\varphi}$ and $\hat{A}^{\neg\varphi}$.

Definition (Monitor $\mathcal{M}^{\varphi}$ for an LTL formula φ)

We define the product automaton $\overline{\mathcal{A}}^{\varphi} = \tilde{A}^{\varphi} \times \tilde{A}^{\neg\varphi}$ as the Moore machine $(\Sigma, \overline{Q}, \overline{q}_0, \mathbb{B}_3, \overline{\delta}, \overline{\lambda})$, where

- $\overline{Q} = \tilde{Q}^{\varphi} \times \tilde{Q}^{\neg\varphi}$,
- $\overline{q}_0 = (\tilde{q}_0^{\varphi}, \tilde{q}_0^{\neg\varphi})$,
- $\overline{\delta}((q, q'), a) = (\tilde{\delta}^{\varphi}(q, a), \tilde{\delta}^{\neg\varphi}(q', a))$ and
- $\overline{\lambda} : \overline{Q} \to \mathbb{B}_3$ with

$$\overline{\lambda}((q, q')) = \begin{cases} \top & \text{if } q' \notin \tilde{F}^{\neg\varphi} \\ \bot & \text{if } q \notin \tilde{F}^{\varphi} \\ ? & \text{if } q \in \tilde{F}^{\varphi} \text{ and } q' \in \tilde{F}^{\neg\varphi}. \end{cases}$$

The monitor $\mathcal{M}^{\varphi}$ of φ is obtained by minimizing $\overline{\mathcal{A}}^{\varphi}$.

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

The Complete Construction

The Construction

LTL	BA	Emptiness per State	NFA

$$\varphi \longrightarrow \mathcal{A}^{\varphi} \longrightarrow \mathcal{F}^{\varphi} \longrightarrow \hat{\mathcal{A}}^{\varphi}$$

LTL_3 Evaluation

$$[\![u \models \varphi]\!]_3 = \begin{cases} \top & \\ \bot & \text{if } u \notin \mathcal{L}(\mathrm{NFA}^{\varphi}) \\ ? & \end{cases}$$

The Complete Construction

The Construction

LTL	BA	Emptiness per State	NFA

$$\varphi \longrightarrow \mathcal{A}^{\varphi} \longrightarrow \mathcal{F}^{\varphi} \longrightarrow \hat{\mathcal{A}}^{\varphi}$$

$\neg\varphi$

LTL_3 Evaluation

$$[\![u \models \varphi]\!]_3 = \begin{cases} \top & \\ \bot & \text{if } u \notin \mathcal{L}(\mathrm{NFA}^{\varphi}) \\ ? & \end{cases}$$

The Complete Construction

The Construction

LTL	BA	Emptiness per State	NFA

$$\varphi \longrightarrow \mathcal{A}^{\varphi} \longrightarrow \mathcal{F}^{\varphi} \longrightarrow \hat{\mathcal{A}}^{\varphi}$$

$$\neg\varphi \rightarrow \mathcal{A}^{\neg\varphi} \longrightarrow \mathcal{F}^{\neg\varphi} \longrightarrow \hat{\mathcal{A}}^{\neg\varphi}$$

LTL_3 Evaluation

$$[\![u \models \varphi]\!]_3 = \begin{cases} \top & \text{if } u \notin \mathcal{L}(\mathrm{NFA}^{\neg\varphi}) \\ \bot & \text{if } u \notin \mathcal{L}(\mathrm{NFA}^{\varphi}) \\ ? & \text{else} \end{cases}$$

The Complete Construction

The Construction

	LTL		BA		Emptiness per State		NFA
$\varphi \nearrow$	φ	$\longrightarrow$	$\mathcal{A}^{\varphi}$	$\longrightarrow$	$\mathcal{F}^{\varphi}$	$\longrightarrow$	$\hat{\mathcal{A}}^{\varphi}$
$\varphi \searrow$	$\neg\varphi$	$\rightarrow$	$\mathcal{A}^{\neg\varphi}$	$\longrightarrow$	$\mathcal{F}^{\neg\varphi}$	$\longrightarrow$	$\hat{\mathcal{A}}^{\neg\varphi}$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

The Complete Construction

The Construction

	LTL	BA	Emptiness per State	NFA	DFA
$\varphi \nearrow$	$\varphi \longrightarrow$	$\mathcal{A}^{\varphi} \longrightarrow$	$\mathcal{F}^{\varphi} \longrightarrow$	$\hat{\mathcal{A}}^{\varphi} \longrightarrow$	$\tilde{\mathcal{A}}^{\varphi}$
$\searrow$	$\neg\varphi \rightarrow$	$\mathcal{A}^{\neg\varphi} \longrightarrow$	$\mathcal{F}^{\neg\varphi} \longrightarrow$	$\hat{\mathcal{A}}^{\neg\varphi} \longrightarrow$	$\tilde{\mathcal{A}}^{\neg\varphi}$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

The Complete Construction

The Construction

LTL	BA	Emptiness per State	NFA	DFA	FSM

$$\varphi \nearrow \varphi \longrightarrow \mathcal{A}^{\varphi} \longrightarrow \mathcal{F}^{\varphi} \longrightarrow \hat{\mathcal{A}}^{\varphi} \longrightarrow \tilde{\mathcal{A}}^{\varphi} \searrow \mathcal{M}^{\varphi}$$

$$\varphi \searrow \neg\varphi \longrightarrow \mathcal{A}^{\neg\varphi} \longrightarrow \mathcal{F}^{\neg\varphi} \longrightarrow \hat{\mathcal{A}}^{\neg\varphi} \longrightarrow \tilde{\mathcal{A}}^{\neg\varphi} \nearrow \mathcal{M}^{\varphi}$$

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Example

LTL $p \,\mathrm{U}\, q$ $\neg(p \,\mathrm{U}\, q)$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

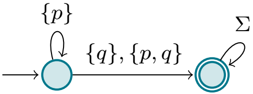
Analysis
Complexity
Monitorable Properties

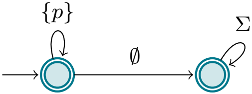
Conclusion

Example

LTL $p \, \mathrm{U} \, q$ $\neg(p \, \mathrm{U} \, q)$

BA





Targets & Outline

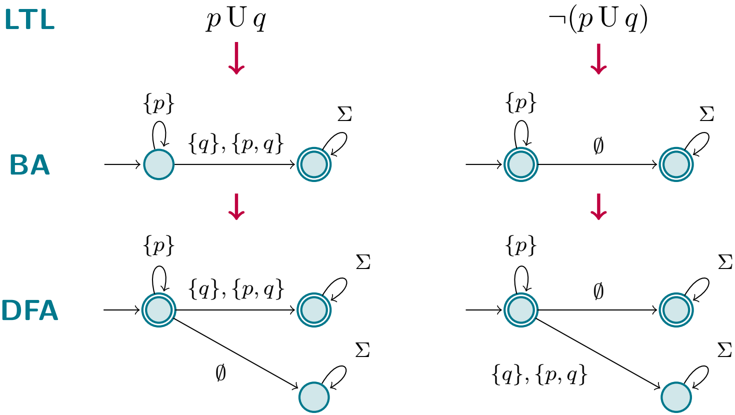
The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Example



Targets & Outline

The Idea

Impartial Anticipation

The Construction

The Construction

From LTL to ABA

Emptiness per State

The Monitor

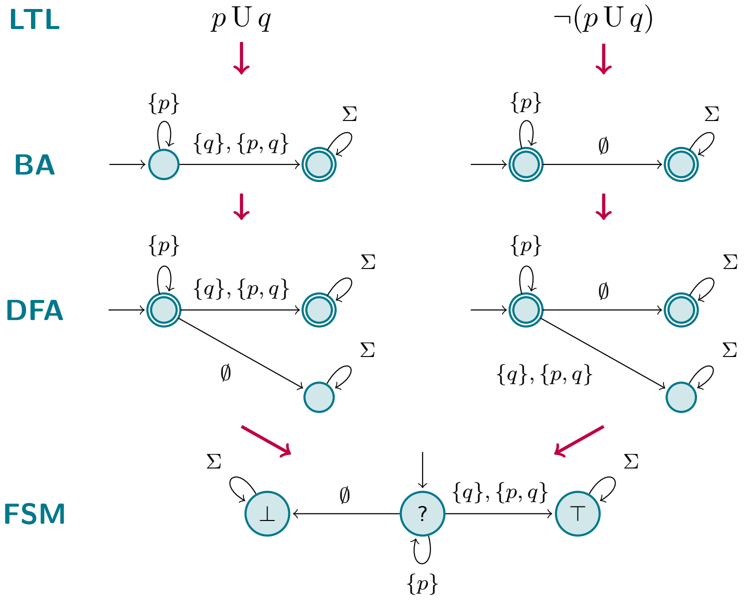
Analysis

Complexity

Monitorable Properties

Conclusion

Example



Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Section

Analysis

Complexity
Monitorable Properties

Chapter 7 “Monitor Construction for Anticipatory Runtime Verification”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

Complexity

The Construction

	LTL	BA	Emptiness per State	NFA	DFA	FSM
$\varphi \nearrow$	φ	$\longrightarrow \mathcal{A}^{\varphi}$	$\longrightarrow \mathcal{F}^{\varphi}$	$\longrightarrow \hat{\mathcal{A}}^{\varphi}$	$\longrightarrow \tilde{\mathcal{A}}^{\varphi}$	$\searrow \mathcal{M}^{\varphi}$
$\varphi \searrow$	$\neg\varphi$	$\longrightarrow \mathcal{A}^{\neg\varphi}$	$\longrightarrow \mathcal{F}^{\neg\varphi}$	$\longrightarrow \hat{\mathcal{A}}^{\neg\varphi}$	$\longrightarrow \tilde{\mathcal{A}}^{\neg\varphi}$	$\nearrow \mathcal{M}^{\varphi}$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

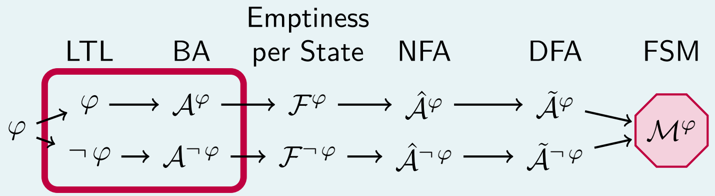
The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Complexity

The Construction



Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

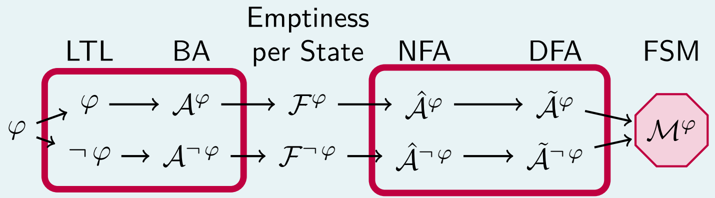
The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Complexity

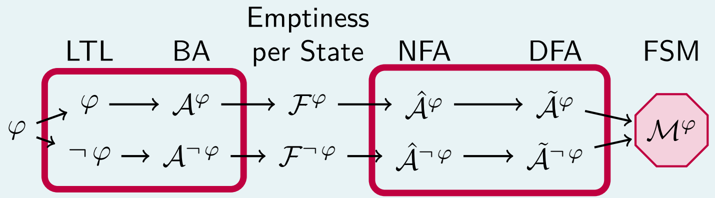
The Construction



Targets & Outline
The Idea
Impartial Anticipation
The Construction
The Construction
From LTL to ABA
Emptiness per State
The Monitor
Analysis
Complexity
Monitorable Properties
Conclusion

Complexity

The Construction



Complexity

$$|M| \in 2^{2^{O(|\varphi|)}}$$

Targets & Outline

The Idea
Impartial Anticipation
The Construction

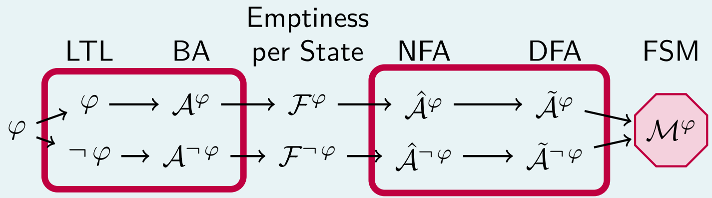
The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Complexity

The Construction



Complexity

$$|M| \in 2^{2^{O(|\varphi|)}}$$

Optimal result!

FSM can be minimised (Myhill-Nerode)

Targets & Outline

The Idea
Impartial Anticipation
The Construction

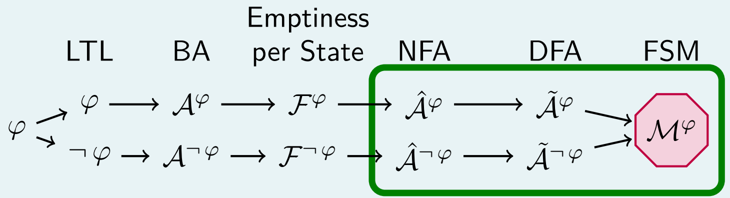
The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

On-the-fly Construction

The Construction



Targets & Outline

The Idea
Impartial Anticipation
The Construction

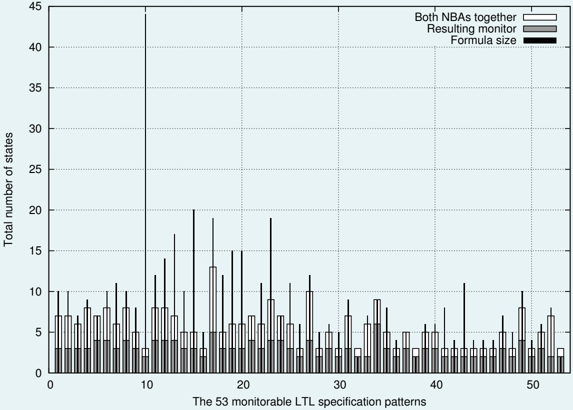
The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Evaluation on Dwyer's Specification Patterns I

Number of States



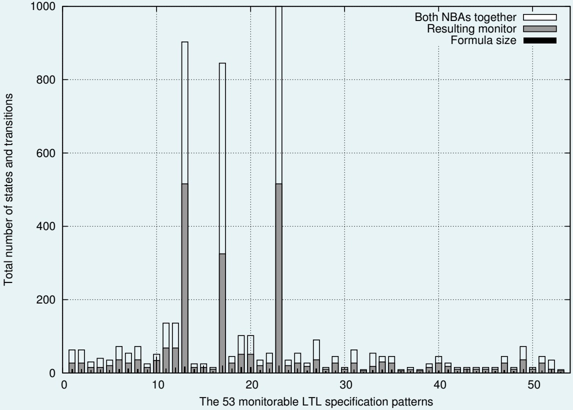
Runtime Verification

M. Leucker & V. Stolz

Targets & Outline
The Idea
Impartial Anticipation
The Construction
The Construction
From LTL to ABA
Emptiness per State
The Monitor
Analysis
Complexity
Monitorable Properties
Conclusion

Evaluation on Dwyer's Specification Patterns II

Total Size



Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Monitorability

When Does Anticipation Help?

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Recall The Good, The Bad and The Ugly

Given a language $L \subseteq \Sigma^\omega$ of infinite words over Σ we call a finite word $u \in \Sigma^*$

- a good prefix for L if $\forall w \in \Sigma^\omega : uw \in L$,
- a bad prefix for L if $\forall w \in \Sigma^\omega : uw \notin L$ and
- an ugly prefix for L if $\forall v \in \Sigma^* : uv$ is neither a good prefix nor a bad prefix.

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Recall The Good, The Bad and The Ugly

Given a language $L \subseteq \Sigma^\omega$ of infinite words over Σ we call a finite word $u \in \Sigma^*$

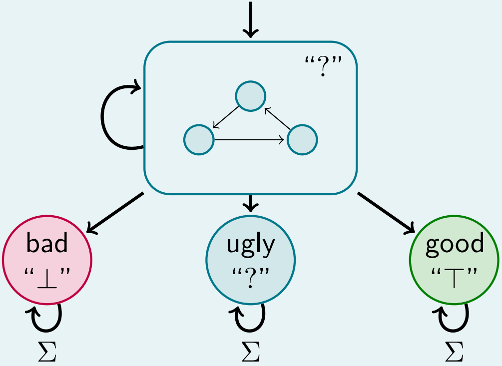
- a good prefix for L if $\forall w \in \Sigma^\omega : uw \in L$,
- a bad prefix for L if $\forall w \in \Sigma^\omega : uw \notin L$ and
- an ugly prefix for L if $\forall v \in \Sigma^* : uv$ is neither a good prefix nor a bad prefix.

LTL_3 indentifies good/bad prefixes:

$$[\![u \models \varphi]\!]_3 = \begin{cases} \top & \text{if } u \text{ is a good prefix for } \mathcal{L}(\varphi) \\ \bot & \text{if } u \text{ is a bad prefix for } \mathcal{L}(\varphi) \\ ? & \text{otherwise.} \end{cases}$$

Monitors Revisited

Structure of Monitors



Targets & Outline

The Idea
Impartial Anticipation
The Construction

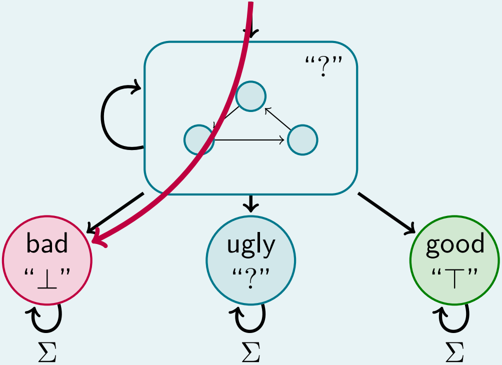
The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Monitors Revisited

Structure of Monitors



Classification of Prefixes of Words

Bad prefixes

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

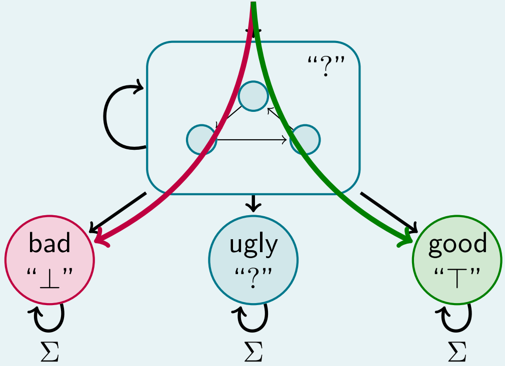
The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Monitors Revisited

Structure of Monitors



Classification of Prefixes of Words

Bad prefixes Good prefixes

Targets & Outline

The Idea
Impartial Anticipation
The Construction

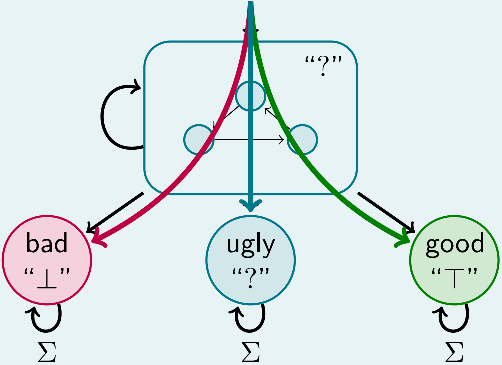
The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

Monitors Revisited

Structure of Monitors



Classification of Prefixes of Words

Bad prefixes Ugly prefixes Good prefixes

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Monitorable

Non-Monitorable

φ is non-monitorable after u, if u cannot be extended to a bad oder good prefix.

Monitorable

φ is monitorable if there is no such u.

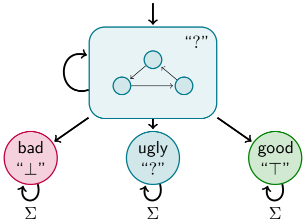
Monitorable

Non-Monitorable

φ is non-monitorable after u, if u cannot be extended to a bad oder good prefix.

Monitorable

φ is monitorable if there is no such u.



Ugly occurs

Consider $\mathrm{G}\,\mathrm{F}\,p$

Targets & Outline

The Idea
- Impartial Anticipation
- The Construction

The Construction
- From LTL to ABA
- Emptiness per State
- The Monitor

Analysis
- Complexity
- Monitorable Properties

Conclusion

Conclusion

1. LTL formulae on infinite words can be translated into alternating Büchi automata.
2. The emptiness per state function computes the satisfiability of an LTL formla and can be used to generate an NFA accepting in the not bad states of the BA of the LTL formula.
3. The impartial anticipatory LTL monitor is based on such an NFA for the LTL formula and one for its complement and identifies good, bad and ugly prefixes.
4. Universalitiy testing for Büchi automata can be avoided by complementing the LTL formula instead of the Büchi automaton.
5. The size of the generated monitor is double-exponential in the size of the LTL formula.

Targets & Outline

The Idea
Impartial Anticipation
The Construction

The Construction
From LTL to ABA
Emptiness per State
The Monitor

Analysis
Complexity
Monitorable Properties

Conclusion

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Chapter 8

LTL with a Predictive Semantics

Course “Runtime Verification”

M. Leucker & V. Stolz

INF5140 / V17

Chapter 8

Learning Targets of Chapter "LTL with a Predictive Semantics".

1. Understand how the underlying program to monitor could be taken into account.
2. Understand how to build a corresponding monitor synthesis procedure.

Chapter 8

Outline of Chapter "LTL with a Predictive Semantics".

Predictive Semantics

- Motivation
- Definition

Monitoring $\mathrm{LTL}_4^{\mathcal{P}}$

Section

Predictive Semantics

Motivation
Definition

Fusing model checking and runtime verification

LTL with a predictive semantics

Recall anticipatory LTL semantics

The truth value of a LTL_3 formula φ with respect to u, denoted by $[\![u \models \varphi]\!]$, is an element of $\mathbb{B}_3$ defined by

$$[\![u \models \varphi]\!] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^\omega : u\sigma \models \varphi \\ \bot & \text{if } \forall \sigma \in \Sigma^\omega : u\sigma \not\models \varphi \\ ? & \text{otherwise.} \end{cases}$$

Applied to the Empty Word

Empty word ε

$$
\begin{aligned}
& [\![\varepsilon \models \varphi]\!]_{\mathcal{P}} = \top \\
\text{iff} \quad & \forall \sigma \in \Sigma^{\omega} \text{ with } \varepsilon\sigma \in \mathcal{P} : \varepsilon\sigma \models \varphi \\
\text{iff} \quad & \mathcal{L}(\mathcal{P}) \models \varphi
\end{aligned}
$$

RV more difficult than MC?

Then runtime verification implicitly answers model checking

Abstraction

An *over-abstraction* or and *over-approximation* of a program $\mathcal{P}$ is a program $\hat{\mathcal{P}}$ such that $\mathcal{L}(\mathcal{P}) \subseteq \mathcal{L}(\hat{\mathcal{P}}) \subseteq \Sigma^\omega$.

Predictive Semantics

Definition (Predictive Semantics of LTL)

Let $\mathcal{P}$ be a program and let $\hat{\mathcal{P}}$ be an over-approximation of $\mathcal{P}$. Let $u \in \Sigma^*$ denote a finite trace. The *truth value* of u and an LTL formula φ with respect to $\hat{\mathcal{P}}$, denoted by $[\![u \models \varphi]\!]_{\hat{\mathcal{P}}} \in \mathbb{B}_4^{¿} = \{\bot, \top, ?, ¿\}$, and defined as follows:

$$[\![u \models \varphi]\!]_{\hat{\mathcal{P}}} = \begin{cases} \top \text{ if} & u \in_\omega \mathcal{L}(\hat{\mathcal{P}}) \wedge \forall w \in \Sigma^\omega : \\ & uw \in \mathcal{L}(\hat{\mathcal{P}}) \Rightarrow [\![uw \models \varphi]\!]_\omega = \top \\ \bot \text{ if} & u \in_\omega \mathcal{L}(\hat{\mathcal{P}}) \wedge \forall w \in \Sigma^\omega : \\ & uw \in \mathcal{L}(\hat{\mathcal{P}}) \Rightarrow [\![uw \models \varphi]\!]_\omega = \bot \\ ? \text{ if} & \exists w, w' \in \Sigma^\omega : uw, uw' \in \mathcal{L}(\hat{\mathcal{P}}) \wedge \\ & [\![uw \models \varphi]\!]_\omega = \top \wedge [\![uw' \models \varphi]\!]_\omega = \bot \\ ¿ \text{ if} & u \notin_\omega \mathcal{L}(\hat{\mathcal{P}}) \end{cases}$$

We use $\mathrm{LTL}_4^{\mathcal{P}}$ to indicate LTL with predictive semantics.

Properties of Predictive Semantics

Remark

Let $\hat{\mathcal{P}}$ be an over-approximation of a program $\mathcal{P}$ over Σ, $u \in \Sigma^*$, and $\varphi \in \mathrm{LTL}$.

- Model checking is more precise than RV with the predictive semantics:
  $$\mathcal{P} \models \varphi \quad \text{implies} \quad [\![u \models \varphi]\!]_{\hat{\mathcal{P}}} \in \{\top, ?\}$$
- RV has no false negatives:
  $$[\![u \models \varphi]\!]_{\hat{\mathcal{P}}} = \bot \quad \text{implies} \quad \mathcal{P} \not\models \varphi$$
- The predictive semantics of an LTL formula is more precise than LTL_3:
  $$[\![u \models \varphi]\!]_3 = \top \quad \text{implies} \quad [\![u \models \varphi]\!]_{\hat{\mathcal{P}}} = \top$$
  $$[\![u \models \varphi]\!]_3 = \bot \quad \text{implies} \quad [\![u \models \varphi]\!]_{\hat{\mathcal{P}}} = \bot$$

The reverse directions are in general not true.

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Section

Monitoring $\mathrm{LTL}_4^{\mathcal{P}}$

Chapter 8 “LTL with a Predictive Semantics”
Course “Runtime Verification”
M. Leucker & V. Stolz
INF5140 / V17

Conclusion

1. $\mathrm{LTL}_4^{\mathcal{P}}$ only considers extensions of the current word leading to executions of an over-abstraction $\hat{\mathcal{P}}$ of the underlying program $\mathcal{P}$.
2. We introduced the new value ¿ of the output alphabet indicating that the current execution has left the over-abstraction.
3. The use of an over-abstraction is the tradeoff between model checking and runtime verification as the use of the program $\mathcal{P}$ itself would implicitly solve the model checking problem.

Chapter 9

Runtime Verification Summary

Course "Runtime Verification"
M. Leucker & V. Stolz
INF5140 / V17

Chapter 9

Learning Targets of Chapter “Runtime Verification Summary”.

1. Understand that FLTL, FLTL_4 and LTL on infinite words share a very similar semantics.
2. Understand that LTL_3 and $\mathrm{LTL}_4^{\mathcal{P}}$ are defined with respect to existing semantics.
3. Understand the difference of propositions and events.

Chapter 9

Outline of Chapter “Runtime Verification Summary”.

Impartial RV

- Common LTL Semantics
- RV on Finite, Terminated Executions
- Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

- LTL on Infinite Words
- Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

- Events vs. Propositions
- Further Topics

Section

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Chapter 9 “Runtime Verification Summary”

Course “Runtime Verification”

M. Leucker & V. Stolz

INF5140 / V17

LTL Semantics

We know the following LTL semantics

- FLTL LTL on finite, completed words
- FLTL_4 impartial LTL on finite, non-completed words
- LTL LTL on infinite words
- LTL_3 anticipatory LTL on finite, non-completed words
- $\mathrm{LTL}_4^{\mathcal{P}}$ anticipatory LTL on finite, non-completed words with respect to an over-abstraction of a program $\mathcal{P}$

- FLTL, FLTL_4 and LTL have very similar semantics with a big common part
- LTL_3 and $\mathrm{LTL}_4^{\mathcal{P}}$ are defined based on LTL

The Common Parts of LTL Semantics

Let AP be a finite set of atomic propositions, $\Sigma = 2^{\mathrm{AP}}$, $p \in \mathrm{AP}$, φ, ψ LTL formulae, and $w \in \Sigma^\infty$ a (finite or infinite) word. The the common part of the LTL semantics FLTL, FLTL_4 and LTL (indicated by $\mathfrak{L}$) of an LTL formula with respect to w is inductively defined as follows:

Boolean Constants

$$[\![w \models \mathrm{true}]\!]_{\mathfrak{L}} = \top$$
$$[\![w \models \mathrm{false}]\!]_{\mathfrak{L}} = \bot$$

Boolean Combinations

$$[\![w \models \neg \varphi]\!]_{\mathfrak{L}} = \overline{[\![w \models \varphi]\!]_{\mathfrak{L}}}$$
$$[\![w \models \varphi \vee \psi]\!]_{\mathfrak{L}} = [\![w \models \varphi]\!]_{\mathfrak{L}} \sqcup [\![w \models \psi]\!]_{\mathfrak{L}}$$
$$[\![w \models \varphi \wedge \psi]\!]_{\mathfrak{L}} = [\![w \models \varphi]\!]_{\mathfrak{L}} \sqcap [\![w \models \psi]\!]_{\mathfrak{L}}$$

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

The Common Parts of LTL Semantics

Let AP be a finite set of atomic propositions, $\Sigma = 2^{\mathrm{AP}}$, $p \in \mathrm{AP}$, φ, ψ LTL formulae, and $w \in \Sigma^\infty$ a (finite or infinite) word. The the common part of the LTL semantics FLTL, FLTL_4 and LTL (indicated by $\mathfrak{L}$) of an LTL formula with respect to w is inductively defined as follows:

Atomic Propositions

$$[\![w \models p]\!]_{\mathfrak{L}} = \begin{cases} \top & \text{if } p \in w_1 \\ \bot & \text{if } p \notin w_1 \end{cases}$$

Local Temporal Operators

$$[\![w \models \mathrm{X}\,\varphi]\!]_{\mathfrak{L}} = \text{defined dependent of } \mathfrak{L} \text{ later}$$

$$[\![w \models \overline{\mathrm{X}}\,\varphi]\!]_{\mathfrak{L}} = \text{defined dependent of } \mathfrak{L} \text{ later}$$

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

The Common Parts of LTL Semantics

Let AP be a finite set of atomic propositions, $\Sigma = 2^{\mathrm{AP}}$, $p \in \mathrm{AP}$, φ, ψ LTL formulae, and $w \in \Sigma^\infty$ a (finite or infinite) word. The the common part of the LTL semantics FLTL, FLTL_4 and LTL (indicated by $\mathfrak{L}$) of an LTL formula with respect to w is inductively defined as follows:

Fixed Point Operators

$$[\![w \models \varphi \, \mathrm{U} \, \psi]\!]_{\mathfrak{L}} = \begin{cases} \top \text{ if } \exists i, 1 \leq i \leq |w| : ([\![w^i \models \psi]\!]_{\mathfrak{L}} = \top \\ \qquad \text{and } \forall k, 1 \leq k < i : [\![w^k \models \varphi]\!]_{\mathfrak{L}} = \top) \\ \text{defined dependent of } \mathfrak{L} \text{ later else} \end{cases}$$

$$[\![w \models \varphi \, \mathrm{R} \, \psi]\!]_{\mathfrak{L}} = \overline{[\![w \models \neg \varphi \, \mathrm{U} \, \neg \psi]\!]_{\mathfrak{L}}}$$

$$[\![w \models \mathrm{F} \, \varphi]\!]_{\mathfrak{L}} = [\![w \models \mathrm{true} \, \mathrm{U} \, \varphi]\!]_{\mathfrak{L}}$$

$$[\![w \models \mathrm{G} \, \varphi]\!]_{\mathfrak{L}} = [\![w \models \mathrm{false} \, \mathrm{R} \, \varphi]\!]_{\mathfrak{L}}$$

Targets & Outline

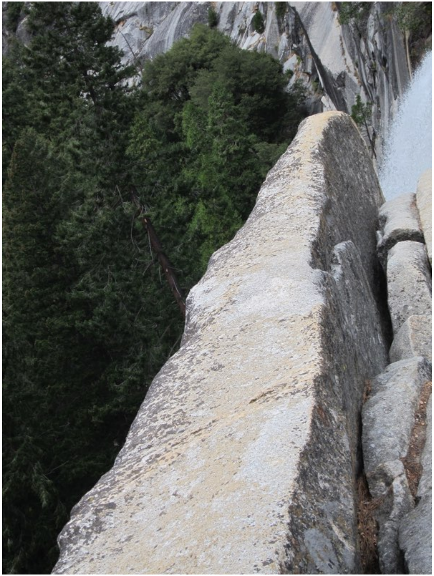
Impartial RV
- Common LTL Semantics
- RV on Finite, Terminated Executions
- Impartial RV on Finite, Non-Terminated Words

Anticipatory RV
- LTL on Infinite Words
- Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics
- Events vs. Propositions
- Further Topics

Conclusion

LTL on Finite, Terminated Words



Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

FLTL: LTL on Finite, Completed Words

Let φ be an LTL formual. We then define the semantics of FLTL by extending the common LTL semantics of an LTL formula with respect to $w \in \Sigma^*$ as follows:

Local Temporal Operators

$$[\![w \models \mathrm{X}\,\varphi]\!]_2 = \begin{cases} [\![w^2 \models \varphi]\!]_2 & \text{if } |w| > 1 \\ \bot & \text{else} \end{cases}$$

$$[\![w \models \overline{\mathrm{X}}\,\varphi]\!]_2 = \begin{cases} [\![w^2 \models \varphi]\!]_2 & \text{if } |w| > 1 \\ \top & \text{else} \end{cases}$$

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

FLTL: LTL on Finite, Completed Words

Let φ be an LTL formual. We then define the semantics of FLTL by extending the common LTL semantics of an LTL formula with respect to $w \in \Sigma^*$ as follows:

Fixed Point Operator Until

$$[\![w \models \varphi \,\mathrm{U}\, \psi]\!]_2 = \begin{cases} \top \text{ if } \exists i, 1 \leq i \leq |w| : ([\![w^i \models \psi]\!]_2 = \top \\ \qquad \text{and } \forall k, 1 \leq k < i : [\![w^k \models \varphi]\!]_2 = \top) \\ \bot \text{ else} \end{cases}$$

Targets & Outline

Impartial RV
- Common LTL Semantics
- RV on Finite, Terminated Executions
- Impartial RV on Finite, Non-Terminated Words

Anticipatory RV
- LTL on Infinite Words
- Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics
- Events vs. Propositions
- Further Topics

Conclusion

Monitor Function For FLTL

The function

$$\mathrm{evlFLTL} : \Sigma^+ \times \mathrm{LTL} \to \mathbb{B}_2$$

takes a finite completed word $w \in \Sigma^+$ and
an LTL formula φ and
returns $[\![w \models \varphi]\!]_2$.

$\mathrm{evlFLTL}$ evaluates recursively

- boolean constants, combinations and atomic propisitions directly,
- the next operator by omitting the first letter of the word and
- the fixed point operators using their fixed point equations.

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

FLTL_4: LTL on Finite, Non-Completed Words

Let φ be an LTL formual. We then define the semantics of FLTL_4 by extending the common LTL semantics of an LTL formula with respect to $w \in \Sigma^*$ as follows:

Local Temporal Operators

$$[\![w \models \mathrm{X}\,\varphi]\!]_4 = \begin{cases} [\![w^2 \models \varphi]\!]_4 & \text{if } |w| > 1 \\ \bot^p & \text{else} \end{cases}$$

$$[\![w \models \overline{\mathrm{X}}\,\varphi]\!]_4 = \begin{cases} [\![w^2 \models \varphi]\!]_4 & \text{if } |w| > 1 \\ \top^p & \text{else} \end{cases}$$

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

FLTL_4: LTL on Finite, Non-Completed Words

Let φ be an LTL formual. We then define the semantics of FLTL_4 by extending the common LTL semantics of an LTL formula with respect to $w \in \Sigma^*$ as follows:

Fixed Point Operator Until

$$
\begin{aligned}
&[\![w \models \varphi \,\mathrm{U}\, \psi]\!]_4 \\
&\quad = \left(\bigsqcup_{1 \leq i \leq |w|} \left([\![w^i \models \psi]\!]_4 \sqcap \prod_{1 \leq j < i} [\![w^j \models \varphi]\!]_4 \right) \right) \\
&\qquad\qquad \sqcup \left(\bot^p \sqcap \prod_{1 \leq i \leq |w|} [\![w^i \models \varphi]\!]_4 \right)
\end{aligned}
$$

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

Monitor Function for FLTL_4

The function

$$\mathrm{evlFLTL}_4 : \Sigma \times \mathrm{LTL} \to \mathbb{B}_4 \times \mathrm{LTL}$$

takes a letter $a \in \Sigma$ of a finite non-completed word and an LTL formula φ and returns $[\![a \models \varphi]\!]_4$ and a new LTL formula φ'.

$\mathrm{evlFLTL}_4$

- is based on the ideas of $\mathrm{evlFLTL}$, but
- performs (not recursive) formula rewriting (progression) and
- can be used as transition function of an AMM.

Monitor For FLTL_4

The monitor AMM $\mathcal{M}_\varphi = (\Sigma, Q, q_0, \Gamma, \delta)$ of the LTL formula φ consists of

- the input alphabet $\Sigma = 2^{\mathrm{AP}}$,
- the states Q containing all subformulae of φ,
- the initial state $q_0 = \varphi$,
- the output alphabet $\Gamma = \mathcal{B}_4 = \{\bot, \bot^p, \top^p, \top\}$ and
- the transition function $\delta = \mathrm{evlFLTL}_4$, where boolean combinations are interpreted over $B^+(Q)$.

Such an AMM can be translated into an MM using conjunctive or disjunctive normal forms as new states.

Section

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Chapter 9 “Runtime Verification Summary”

Course “Runtime Verification”

M. Leucker & V. Stolz

INF5140 / V17

LTL: LTL on Infinite Words

Let φ be an LTL formual. We then define the semantics LTL by extending the common LTL semantics of an LTL formula with respect to $w \in \Sigma^\omega$ as follows:

Local Temporal Operators

$$[\![w \models \mathrm{X}\,\varphi]\!]_\omega = [\![w \models \overline{\mathrm{X}}\,\varphi]\!]_\omega = [\![w^2 \models \varphi]\!]_\omega$$

LTL: LTL on Infinite Words

Let φ be an LTL formual. We then define the semantics LTL by extending the common LTL semantics of an LTL formula with respect to $w \in \Sigma^\omega$ as follows:

Fixed Point Operator Until

$$[\![w \models \varphi \,\mathrm{U}\, \psi]\!]_\omega = \begin{cases} \top \text{ if } \exists i, 1 \leq i : ([\![w^i \models \psi]\!]_\omega = \top \\ \qquad \text{and } \forall k, 1 \leq k < i : [\![w^k \models \varphi]\!]_\omega = \top) \\ \bot \text{ else} \end{cases}$$

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

Monitor For LTL

The monitor ABA $\mathcal{A}^\varphi = (\Sigma, Q, q_0, \delta, F)$ of the LTL formula φ consists of

- the input alphabet $\Sigma = 2^{\mathrm{AP}}$,
- the states Q containing all subformulae of φ,
- the initial state $q_0 = \varphi$,
- the transition function δ, that performs progression like $\mathrm{evlFLTL}_4$, and
- the set F of accepting states, that contains all subformulae searching a greatest fixpoint.

Such an ABA can be translated into an BA using a power set construction where every state consists of two sets of states: All states from paths where we already saw an accepting states and states from paths where we still need to see an accepting state.

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

LTL_3: LTL on Finite, Non-Completed Words

Let φ be an LTL formual. We then define the semantics LTL_3 of an LTL formula with respect to $w \in \Sigma^*$ based on the LTL semantics as follows:

$$[\![u \models \varphi]\!]_3 = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^\omega : [\![w\sigma \models \varphi]\!]_\omega = \top \\ \bot & \text{if } \forall \sigma \in \Sigma^\omega : [\![w\sigma \models \varphi]\!]_\omega = \bot \\ ? & \text{else.} \end{cases}$$

Targets & Outline

Impartial RV

Common LTL Semantics

RV on Finite, Terminated Executions

Impartial RV on Finite, Non-Terminated Words

Anticipatory RV

LTL on Infinite Words

Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics

Events vs. Propositions

Further Topics

Conclusion

Monitor For LTL_3

The monitor FSM $\mathcal{M}_\varphi = \tilde{\mathcal{A}}^\varphi \times \tilde{\mathcal{A}}^{\neg\varphi}$ of the LTL formula φ consists of

- the DFA $\tilde{\mathcal{A}}^\varphi$ computed from the LTL monitor $\mathcal{A}^\varphi = (\Sigma, Q^\varphi, \varphi, \delta^\varphi, F^\varphi)$ via the emptiness per state function,
- the DFA $\tilde{\mathcal{A}}^{\neg\varphi}$ computed from the LTL monitor $\mathcal{A}^{\neg\varphi} = (\Sigma, Q^{\neg\varphi}, \neg\varphi, \delta^{\neg\varphi}, F^{\neg\varphi})$ via the emptiness per state function and
- the labeling function $\lambda : Q \to \mathbb{B}_3$ that prints
  - $\top$ if $\tilde{\mathcal{A}}^\varphi$ is in a rejecting state
  - $\bot$ if $\tilde{\mathcal{A}}^{\neg\varphi}$ is in a rejecting state
  - $?$ if both are in accepting states.

Targets & Outline
Impartial RV
Common LTL Semantics
RV on Finite, Terminated Executions
Impartial RV on Finite, Non-Terminated Words
Anticipatory RV
LTL on Infinite Words
Anticipatory RV on Finite, Non-Terminated Words
Other LTL Semantics
Events vs. Propositions
Further Topics
Conclusion

$\mathrm{LTL}_4^{\mathcal{P}}$: Predictive LTL on Finite, Non-Completed Words

Let φ be an LTL formual and $\mathcal{P}$ a program. We then define the semantics $\mathrm{LTL}_3^{\mathcal{P}}$ of an LTL formula with respect to $w \in \Sigma^*$ and and an over-approximation $\hat{\mathcal{P}}$ of $\mathcal{P}$ based on the LTL semantics as follows:

$$
[\![u \models \varphi]\!]_{\hat{\mathcal{P}}} = \begin{cases} \top \text{ if} & u \in_\omega \mathcal{L}(\hat{\mathcal{P}}) \wedge \forall w \in \Sigma^\omega : \\ & uw \in \mathcal{L}(\hat{\mathcal{P}}) \Rightarrow [\![uw \models \varphi]\!]_\omega = \top \\ \bot \text{ if} & u \in_\omega \mathcal{L}(\hat{\mathcal{P}}) \wedge \forall w \in \Sigma^\omega : \\ & uw \in \mathcal{L}(\hat{\mathcal{P}}) \Rightarrow [\![uw \models \varphi]\!]_\omega = \bot \\ ? \text{ if} & \exists w, w' \in \Sigma^\omega : uw, uw' \in \mathcal{L}(\hat{\mathcal{P}}) \wedge \\ & [\![uw \models \varphi]\!]_\omega = \top \wedge [\![uw' \models \varphi]\!]_\omega = \bot \\ ¿ \text{ if} & u \notin_\omega \mathcal{L}(\hat{\mathcal{P}}) \end{cases}
$$

Section

Other LTL Semantics

Events vs. Propositions
Further Topics

LTL With Propositions

So far we always used

- a set AP of atomic propositions and
- an alphabet $\Sigma = 2^{\mathrm{AP}}$.

An LTL formula consisting of an atomic proposition $p \in \mathrm{AP}$ gets evaluated with respect to a word $w \in \Sigma^\infty$ as follows:

$$[\![w \models p]\!]_{\mathfrak{L}} = \begin{cases} \top & \text{if } p \in w_1 \\ \bot & \text{if } p \notin w_1 \end{cases}$$

LTL With Events

We now consider

- a set EV of events and
- an alphabet $\Sigma = \mathrm{EV}$.

An LTL formula consisting of an event $e \in \mathrm{EV}$ then gets evaluated with respect to a word $w \in \Sigma^\infty$ as follows:

$$[\![w \models e]\!]_{\mathfrak{L}} = \begin{cases} \top & \text{if } e = w_1 \\ \bot & \text{if } e \neq w_1 \end{cases}$$

Targets & Outline

Impartial RV
Common LTL Semantics
RV on Finite, Terminated Executions
Impartial RV on Finite, Non-Terminated Words

Anticipatory RV
LTL on Infinite Words
Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics
Events vs. Propositions
Further Topics

Conclusion

Propositions vs. Events

Propositions

- A state consist of a set of propisitions.
- A word $w \in (2^{\mathrm{AP}})^\omega$ is a sequence of states.
- The formula $p \wedge q$ requires that p and q hold in the current state and therefore can be fulfilled.

Events

- A state consist of one event.
- A word $w \in (\mathrm{EV})^\omega$ is a sequence of events.
- The formula $p \wedge q$ requires that the current state is p and q and therefore cannot be fulfilled for $p \neq q$!

Targets & Outline

Impartial RV
Common LTL Semantics
RV on Finite, Terminated Executions
Impartial RV on Finite, Non-Terminated Words

Anticipatory RV
LTL on Infinite Words
Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics
Events vs. Propositions
Further Topics

Conclusion

LTL With Past

Sometimes it makes things easier to look back

Consider the property "Every alarm is due to a fault" expressed in LTL as follows:

$$\mathrm{fault}\,\mathrm{R}(\neg\,\mathrm{alarm} \vee \mathrm{fault})$$

Using the past operator once (finally in past) O this can be expressed more intuitive as follows:

$$\mathrm{G}(\mathrm{alarm} \rightarrow \mathrm{O}\,\mathrm{fault})$$

- Monitor Generation for LTL with past uses two-way automata.
- LTL with past is kind of syntactic sugar as it is not more expressive than future LTL.

Targets & Outline

Impartial RV
- Common LTL Semantics
- RV on Finite, Terminated Executions
- Impartial RV on Finite, Non-Terminated Words

Anticipatory RV
- LTL on Infinite Words
- Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics
- Events vs. Propositions
- Further Topics

Conclusion

Regular LTL

Adding the Power of Regular Expressions to the Elegance of LTL

Consider the property "p holds in every other state" for a proposition $p \in \mathrm{AP}$ expressed as language as follows:

$$(\Sigma \circ \{p\})^*$$

- LTL can express exactly the star-free languages.
- This property cannot be expressed in LTL.

The property "φ holds in every other state" for an RLTL formula φ can be expressed as

$$\varphi|(\Sigma \circ \Sigma)\rangle\emptyset$$

using the ternary weak power operator $\bullet|\bullet\rangle\bullet$ with a delay of two states expressed as language $\Sigma \circ \Sigma$.

Parameterized LTL

Consider the LTL formula $\mathrm{G}(\texttt{close} \rightarrow \mathrm{X}\,\mathrm{G}(\neg\,\texttt{write}))$ that prohibits writing to the closed resource.

Such formula would fail on an execution with two (or more) resources `c1` and `c2`:

```
open(c1); open(c2);
write(c2);
close(c2);
write(c1); // fail
close(c1);
```

We could solve this problem by allowing free variables in LTL formulas. For example $\mathrm{G}(\texttt{close}(c) \rightarrow \mathrm{X}\,\mathrm{G}(\neg\,\texttt{write}(c)))$ is parametric in c.

Targets & Outline

Impartial RV
- Common LTL Semantics
- RV on Finite, Terminated Executions
- Impartial RV on Finite, Non-Terminated Words

Anticipatory RV
- LTL on Infinite Words
- Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics
- Events vs. Propositions
- Further Topics

Conclusion

LTL With Modulo Constraints

Consider a set VAR of integer variables. We then define LTL semantics with respect to infinite sequences of valuations for VAR taking their values in $\mathbb{Z}/n\mathbb{Z}$.

Example

$$\varphi := \mathrm{G}(\mathrm{X}\, x = x)$$

requires that $x \in \mathrm{VAR}$ evaulates in every state to the same value as in the next state (in all states).

Models of such LTL formulas are words $w \in (\mathbb{Z}/n\mathbb{Z})^\omega$:

- $(12)^\omega \models \varphi$,
  because x is always 12.
- $(12; 13)^\omega \not\models \varphi$,
  because x alternates between 12 and 13.

Targets & Outline

Impartial RV
Common LTL Semantics
RV on Finite, Terminated Executions
Impartial RV on Finite, Non-Terminated Words

Anticipatory RV
LTL on Infinite Words
Anticipatory RV on Finite, Non-Terminated Words

Other LTL Semantics
Events vs. Propositions
Further Topics

Conclusion

Conclusion

1. FLTL, FLTL_4 and LTL share common semantics for boolean constants, boolean combinations, atomic propositions and fixed point operators defined using dualization or simplification.
2. We only need to define the semantics of next, weak next and until to specify the semantics of FLTL, FLTL_4 and LTL using the common semantics.
3. LTL_3 and $\mathrm{LTL}_4^{\mathcal{P}}$ are defined based on LTL.
4. Sometimes it make sense to define LTL semantics using events as states instead of sets of propositions.
5. There are very many extensions of LTL and runtime verification not covered (and many not even mentioned) in this course.