

Computation Tree Logic

Luis Tertulino & Ronaldo Silveira

October 23, 2015

- 1 In previous chapters...
- 2 Introduction
- 3 How to communicate
 - Syntax of CTL
 - Semantics of CTL
- 4 Some examples of what we can say
- 5 More about semantics
 - Equivalences
- 6 Improving our language
- 7 Model checking algorithms

Previously on Temporal Logic Week...

Temporal Logic

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- A brief introduction to Propositional Logic, its syntax and its semantics

Previously on Temporal Logic Week...

Temporal Logic

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- A brief introduction to Propositional Logic, its syntax and its semantics
- Formal models of time

Previously on Temporal Logic Week...

Temporal Logic

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- A brief introduction to Propositional Logic, its syntax and its semantics
- Formal models of time
 - Frames and Flows of time

Previously on Temporal Logic Week...

Temporal Logic

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- A brief introduction to Propositional Logic, its syntax and its semantics
- Formal models of time
 - Frames and Flows of time
- Temporal Logic extends the Propositional Logic
 - The connectives H and G

Previously on Temporal Logic Week...

Temporal Logic

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- A brief introduction to Propositional Logic, its syntax and its semantics
- Formal models of time
 - Frames and Flows of time
- Temporal Logic extends the Propositional Logic
 - The connectives H and G
- Some practical applications

Previously on Temporal Logic Week...

Model checking

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

■ “What good is Temporal Logic?”

Previously on Temporal Logic Week...

Model checking

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “What good is Temporal Logic?”
 - Answer: “Temporal Logic is a good method for specifying and reasoning about a concurrent program”.

Previously on Temporal Logic Week...

Model checking

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “What good is Temporal Logic?”
 - Answer: “Temporal Logic is a good method for specifying and reasoning about a concurrent program”.
- Some mathematical definitions about states of a program and (concurrent) programs

Previously on Temporal Logic Week...

Model checking

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “What good is Temporal Logic?”
 - Answer: “Temporal Logic is a good method for specifying and reasoning about a concurrent program”.
- Some mathematical definitions about states of a program and (concurrent) programs
- The Peterson’s algorithm

Previously on Temporal Logic Week...

Model checking

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “What good is Temporal Logic?”
 - Answer: “Temporal Logic is a good method for specifying and reasoning about a concurrent program”.
- Some mathematical definitions about states of a program and (concurrent) programs
- The Peterson’s algorithm
- How the model checking works?

Previously on Temporal Logic Week...

Model checking

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “What good is Temporal Logic?”
 - Answer: “Temporal Logic is a good method for specifying and reasoning about a concurrent program”.
- Some mathematical definitions about states of a program and (concurrent) programs
- The Peterson’s algorithm
- How the model checking works?
- Some strengths and weaknesses of model checking

Previously on Temporal Logic Week...

Linear Temporal Logic

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

■ Syntax and Semantics of LTL

Previously on Temporal Logic Week...

Linear Temporal Logic

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Syntax and Semantics of LTL
- ω -languages, Kripke structures, paths and traces
- Buchi automata and LTL model checking

Motivation

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Needing of expressing uncertainty;

Motivation

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Needing of expressing uncertainty;
- Different paths of the future;

Intuition

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

In Computation Tree Logic (CTL) the model of time is a tree-like structure. This way, we cannot use Linear Temporal Logic (LTL) to express the existence of a certain path of time in which some event occurs.

History

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

CTL was defined by:



Figure 1:
Mordechai Ben-Ari



Figure 2: Amir
Pnueli



Figure 3: Zohar
Manna

History

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

And, at the same time by:



Figure 4: Ernest
Allen Emerson



Figure 5: Edmund
Clarke

Syntax

Definition

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

The syntax of CTL consists on the syntax of temporal logic plus some path operators. The class of formulas can be defined in Backus-Naur form. If ϕ is a formula:

Syntax

Definition

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

The syntax of CTL consists on the syntax of temporal logic plus some path operators. The class of formulas can be defined in Backus-Naur form. If ϕ is a formula:

$$\phi ::= \perp \mid \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid AX\phi \mid EX\phi \mid$$

$$AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A[\phi U\phi] \mid E[\phi U\phi]$$

Syntax

Definition

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

The syntax of CTL consists on the syntax of temporal logic plus some path operators. The class of formulas can be defined in Backus-Naur form. If ϕ is a formula:

$$\phi ::= \perp \mid \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid AX\phi \mid EX\phi \mid$$

$$AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A[\phi U\phi] \mid E[\phi U\phi]$$

With p as a literal (atomic formula), AX , EX , AF , EF , AG e EG unary operators.

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

Syntax

Intuition

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

Syntax

Intuition

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silva

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- A : is the universal quantifier over paths. Read as: “in all possible paths”;

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- A : is the universal quantifier over paths. Read as: “in all possible paths”;
- E : is the existential quantifier over paths. Read as: “exists a path in which”;

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- A : is the universal quantifier over paths. Read as: “in all possible paths”;
- E : is the existential quantifier over paths. Read as: “exists a path in which”;

The temporal operators, as in LTL, can be read as:

- X : “in the next state”;

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- A : is the universal quantifier over paths. Read as: “in all possible paths”;
- E : is the existential quantifier over paths. Read as: “exists a path in which”;

The temporal operators, as in LTL, can be read as:

- X : “in the next state”;
- F : “There is some state in the future (eventually)”;

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- A : is the universal quantifier over paths. Read as: “in all possible paths”;
- E : is the existential quantifier over paths. Read as: “exists a path in which”;

The temporal operators, as in LTL, can be read as:

- X : “in the next state”;
- F : “There is some state in the future (eventually)”;
- G : “Globally (in all future states)”;

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- A : is the universal quantifier over paths. Read as: “in all possible paths”;
- E : is the existential quantifier over paths. Read as: “exists a path in which”;

The temporal operators, as in LTL, can be read as:

- X : “in the next state”;
- F “There is some state in the future (eventually)”;
- G “Globally (in all future states)”;
- $\varphi U \psi$: φ is true at least until ψ becomes true;

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Notice that, in CTL, the combination of path specific operators and temporal operators are atomic, e.g., AF is an atomic operator that can be read as “In all paths in the future there is some state where...”;

- Notice that, in CTL, the combination of path specific operators and temporal operators are atomic, e.g., AF is an atomic operator that can be read as “In all paths in the future there is some state where...”;
- Notice as well that the binary operators $A[\varphi U \psi]$ and $E[\varphi U \psi]$ can be represented as AU and EU , respectively;

- Notice that, in CTL, the combination of path specific operators and temporal operators are atomic, e.g., AF is an atomic operator that can be read as “In all paths in the future there is some state where...”;
- Notice as well that the binary operators $A[\varphi U \psi]$ and $E[\varphi U \psi]$ can be represented as AU and EU , respectively;
- We assume that, similarly to the \neg operator, the “new” unary operators (AX , EX , AF , EF , AG , and EG) have the first precedence. Next comes the \wedge and \vee operators. And at last the \rightarrow , AU and EU ;

Examples

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL

Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

■ Examples of well-formed formulas:

- $AG(p \vee EFq)$

Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

■ Examples of well-formed formulas:

- $AG(p \vee EFq)$
- $AX(q \rightarrow E[(p \vee q)Ur])$

Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

■ Examples of well-formed formulas:

- $AG(p \vee EFq)$
- $AX(q \rightarrow E[(p \vee q)Ur])$
- $EFEGp \rightarrow AFr$ Note that this is binded as
 $(EFEGp) \rightarrow AFr$, not as $EFEG(p \rightarrow AFr)$

Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

■ Examples of well-formed formulas:

- $AG(p \vee EFq)$
- $AX(q \rightarrow E[(p \vee q)Ur])$
- $EFEGp \rightarrow AFr$ Note that this is binded as $(EFEGp) \rightarrow AFr$, not as $EFEG(p \rightarrow AFr)$

■ Example of formulas that are not well-formed:

- $A\neg G\neg p$

Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

■ Examples of well-formed formulas:

- $AG(p \vee EFq)$
- $AX(q \rightarrow E[(p \vee q)Ur])$
- $EFEGp \rightarrow AFr$ Note that this is binded as $(EFEGp) \rightarrow AFr$, not as $EFEG(p \rightarrow AFr)$

■ Example of formulas that are not well-formed:

- $A\neg G\neg p$
- $F[pUs]$

Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Examples of well-formed formulas:
 - $AG(p \vee EFq)$
 - $AX(q \rightarrow E[(p \vee q)Ur])$
 - $EFEGp \rightarrow AFr$ Note that this is binded as $(EFEGp) \rightarrow AFr$, not as $EFEG(p \rightarrow AFr)$
- Example of formulas that are not well-formed:
 - $A\neg G\neg p$
 - $F[pUs]$
 - $A[pUs \wedge qUs]$

Semantics

Intuition of semantics

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

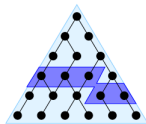
Some examples of
what we can say

More about
semantics
Equivalences

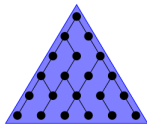
Improving our
language

Model checking
algorithms

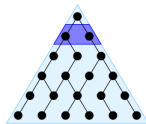
References



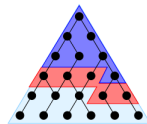
AF p



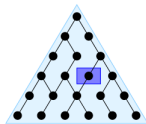
AG p



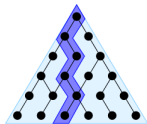
AX p



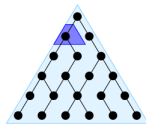
A [p **U** q]



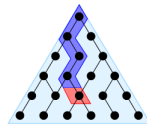
EF p



EG p



EX p



E [p **U** q]

Semantics

Definition of model

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Definition

Let *Atoms* be a set of atomic formulas. A **transition system** or **model** \mathcal{M} is a triple $\mathcal{M} = (S, \rightarrow, L)$ in which S is a set of states, \rightarrow is a binary relation over S ($\rightarrow \subseteq S \times S$) such that for every state $s \in S$, exists a s' that $s \rightarrow s'$ and $L : S \rightarrow \mathcal{P}(\text{Atoms})$ (or $L : S \rightarrow (\text{Atoms} \rightarrow \{0, 1\})$) is a labelling function.

Semantics

Definition of model

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Definition

Let *Atoms* be a set of atomic formulas. A **transition system** or **model** \mathcal{M} is a triple $\mathcal{M} = (S, \rightarrow, L)$ in which S is a set of states, \rightarrow is a binary relation over S ($\rightarrow \subseteq S \times S$) such that for every state $s \in S$, exists a s' that $s \rightarrow s'$ and $L : S \rightarrow \mathcal{P}(\text{Atoms})$ (or $L : S \rightarrow (\text{Atoms} \rightarrow \{0, 1\})$) is a labelling function.

CTL formulas are satisfied by a transition system and a specific state.

Notation: we will use $\mathcal{M}, s \models \varphi$ to denote that the model \mathcal{M}, s satisfies the formula φ

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
**Semantics of
CTL**

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Definition

The **satisfaction** of a formula in CTL is recursive over the structure of the formula. It can be done as follows:

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \perp$ for all $s \in S$

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \perp$ for all $s \in S$
- $\mathcal{M}, s \models p$ iff $p \in L(s)$

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate

Syntax of CTL

Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \perp$ for all $s \in S$
- $\mathcal{M}, s \models p$ iff $p \in L(s)$
- $\mathcal{M}, s \models \neg\varphi$ iff $\mathcal{M}, s \not\models \varphi$

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \perp$ for all $s \in S$
- $\mathcal{M}, s \models p$ iff $p \in L(s)$
- $\mathcal{M}, s \models \neg\varphi$ iff $\mathcal{M}, s \not\models \varphi$
- $\mathcal{M}, s \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, s \models \varphi_1$ AND $\mathcal{M}, s \models \varphi_2$

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \perp$ for all $s \in S$
- $\mathcal{M}, s \models p$ iff $p \in L(s)$
- $\mathcal{M}, s \models \neg\varphi$ iff $\mathcal{M}, s \not\models \varphi$
- $\mathcal{M}, s \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, s \models \varphi_1$ AND $\mathcal{M}, s \models \varphi_2$
- $\mathcal{M}, s \models \varphi_1 \vee \varphi_2$ iff $\mathcal{M}, s \models \varphi_1$ OR $\mathcal{M}, s \models \varphi_2$

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \perp$ for all $s \in S$
- $\mathcal{M}, s \models p$ iff $p \in L(s)$
- $\mathcal{M}, s \models \neg\varphi$ iff $\mathcal{M}, s \not\models \varphi$
- $\mathcal{M}, s \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, s \models \varphi_1$ AND $\mathcal{M}, s \models \varphi_2$
- $\mathcal{M}, s \models \varphi_1 \vee \varphi_2$ iff $\mathcal{M}, s \models \varphi_1$ OR $\mathcal{M}, s \models \varphi_2$
- $\mathcal{M}, s \models \varphi_1 \rightarrow \varphi_2$ iff $\mathcal{M}, s \not\models \varphi_1$ OR $\mathcal{M}, s \models \varphi_2$

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models AX\varphi$ iff for all s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \models \varphi$.
Thus, AX says: “in every next state...”

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models AX\varphi$ iff for all s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \models \varphi$.
Thus, AX says: “in every next state...”
- $\mathcal{M}, s \models EX\varphi$ iff exists s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \models \varphi$. Thus,
 EX says: “in some next state...”

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models AX\varphi$ iff for all s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \models \varphi$.
Thus, AX says: “in every next state...”
- $\mathcal{M}, s \models EX\varphi$ iff exists s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \models \varphi$. Thus,
 EX says: “in some next state...”
- $\mathcal{M}, s \models AG\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which
 $s = s_1$, for all s_i , $\mathcal{M}, s_i \models \varphi$. Thus, AG says: “In all
possible paths from now on in all next states...”

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models AX\varphi$ iff for all s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \models \varphi$.
Thus, AX says: “in every next state...”
- $\mathcal{M}, s \models EX\varphi$ iff exists s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \models \varphi$. Thus,
 EX says: “in some next state...”
- $\mathcal{M}, s \models AG\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which
 $s = s_1$, for all s_i , $\mathcal{M}, s_i \models \varphi$. Thus, AG says: “In all
possible paths from now on in all next states...”
- $\mathcal{M}, s \models EG\varphi$ iff exists some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in
which $s = s_1$, for all s_i , $\mathcal{M}, s_i \models \varphi$. Thus, EG says: “Exists
a path from now on in all next states...”

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s, \models AF\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which $s = s_1$, exists $s_i, \mathcal{M}, s_i \models \varphi$. Thus, AF says: “In all possible paths from now on, in some next state...”

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s, \models AF\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which $s = s_1$, exists $s_i, \mathcal{M}, s_i \models \varphi$. Thus, AF says: “In all possible paths from now on, in some next state...”
- $\mathcal{M}, s, \models EF\varphi$ iff exists some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which $s = s_1$, that exists $s_i, \mathcal{M}, s_i \models \varphi$. Thus, EF says: “In some path from now on, in some next state...”

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s, \models AF\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which $s = s_1$, exists $s_i, \mathcal{M}, s_i \models \varphi$. Thus, AF says: “In all possible paths from now on, in some next state...”
- $\mathcal{M}, s, \models EF\varphi$ iff exists some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which $s = s_1$, that exists $s_i, \mathcal{M}, s_i \models \varphi$. Thus, EF says: “In some path from now on, in some next state...”
- $\mathcal{M}, s, \models A[\varphi_1 U \varphi_2]$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which $s = s_1$, this path satisfies $\varphi_1 U \varphi_2$, i.e., exists s_i in the path such that $\mathcal{M}, s_i \models \varphi_2$ and, for all $j < i$, $\mathcal{M}, s_j \models \varphi_1$. Thus, AU says: “For all paths from now on, until some state...”

Semantics

Satisfaction

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Take an arbitrary model \mathcal{M} . Let s, s_1, s_2, s_3 be states in S . Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models E[\varphi_1 U \varphi_2]$ iff exists some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$ in which $s = s_1$, this path satisfies $\varphi_1 U \varphi_2$. Thus, *EU* says: “In some path from now on, until some state...”

Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “It’s possible to get to a state where something has started but it’s not ready”: $EF(started \wedge \neg ready)$

Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “It’s possible to get to a state where something has started but it’s not ready”: $EF(started \wedge \neg ready)$
- “A certain process is enabled infinitely often on every computation path”: $AG(AF enabled)$

Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “It’s possible to get to a state where something has started but it’s not ready”: $EF(started \wedge \neg ready)$
- “A certain process is enabled infinitely often on every computation path”: $AG(Enabled)$
- “An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor”:
 $AG(floor2 \wedge directionup \wedge button5 \rightarrow A[directionup U floor5])$

Examples

Finite state automata

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

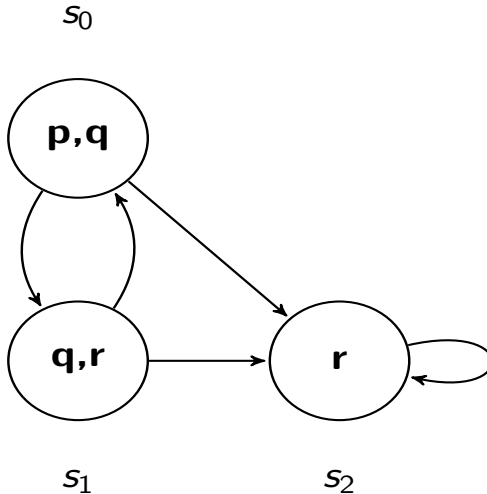
Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References



Examples

Corresponding tree

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

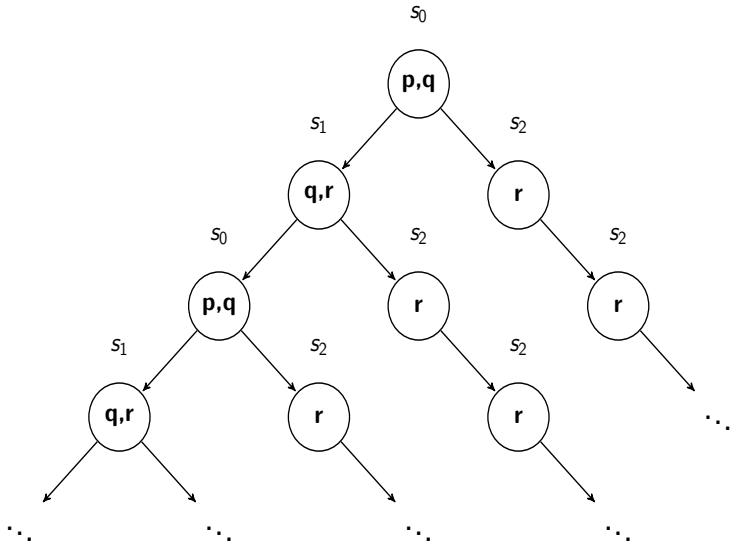
Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References



Examples

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Example of formulas that are satisfied by that model:

$$\blacksquare \mathcal{M}, s_0 \models p \wedge q$$

$$\blacksquare \mathcal{M}, s_0 \models \neg r$$

$$\blacksquare \mathcal{M}, s_0 \models EX(q \wedge r)$$

$$\blacksquare \mathcal{M}, s_0 \models \neg AX(q \wedge r)$$

$$\blacksquare \mathcal{M}, s_0 \models \neg EF(p \wedge q)$$

$$\blacksquare \mathcal{M}, s_2 \models EGr$$

$$\blacksquare \mathcal{M}, s_0 \models AFr$$

$$\blacksquare \mathcal{M}, s_0 \models E[(p \wedge q)Ur]$$

$$\blacksquare \mathcal{M}, s_0 \models A[pUr]$$

$$\blacksquare \mathcal{M}, s_0 \models AG(p \vee q \vee r \rightarrow EFEGr)$$

Equivalences

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Definition

Two CTL formulas φ and ψ are said to be **semantically equivalent** if any state in any model which satisfies one of them also satisfies the other.

Equivalences

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Definition

Two CTL formulas φ and ψ are said to be **semantically equivalent** if any state in any model which satisfies one of them also satisfies the other.

Notation: we denote the semantic equivalence of φ and ψ by
$$\varphi \equiv \psi$$

Example of equivalences

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics

Equivalences

Improving our
language

Model checking
algorithms

References

Let φ be an arbitrary CTL formula.

$$\blacksquare \neg AF\varphi \equiv EG\neg\varphi$$

Example of equivalences

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics

Equivalences

Improving our
language

Model checking
algorithms

References

Let φ be an arbitrary CTL formula.

$$\blacksquare \neg AF\varphi \equiv EG\neg\varphi$$

$$\blacksquare \neg EF\varphi \equiv AG\neg\varphi$$

Example of equivalences

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics

Equivalences

Improving our
language

Model checking
algorithms

References

Let φ be an arbitrary CTL formula.

$$\blacksquare \neg AF\varphi \equiv EG\neg\varphi$$

$$\blacksquare \neg EF\varphi \equiv AG\neg\varphi$$

$$\blacksquare \neg AX\varphi \equiv EX\neg\varphi$$

Example of equivalences

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Let φ be an arbitrary CTL formula.

$$\blacksquare \neg AF\varphi \equiv EG\neg\varphi$$

$$\blacksquare \neg EF\varphi \equiv AG\neg\varphi$$

$$\blacksquare \neg AX\varphi \equiv EX\neg\varphi$$

$$\blacksquare AF\varphi \equiv A[\top U \varphi]$$

Example of equivalences

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Let φ be an arbitrary CTL formula.

$$\blacksquare \neg AF\varphi \equiv EG\neg\varphi$$

$$\blacksquare \neg EF\varphi \equiv AG\neg\varphi$$

$$\blacksquare \neg AX\varphi \equiv EX\neg\varphi$$

$$\blacksquare AF\varphi \equiv A[\top U\varphi]$$

$$\blacksquare EF\varphi \equiv E[\top U\varphi]$$

Minimum set of CTL connectives

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Because of the equivalences shown and the ones in propositional logic, we can have some minimum sets of connectives for the CTL syntax. One of them is defined in Backus-Naur formalism below:

$$\phi ::= \perp \mid p \mid \neg\phi \mid \phi \wedge \phi \mid EX\phi \mid AF\phi \mid E[\phi U \phi]$$

That's all we need?

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Even if CTL allow explicit quantification over paths, it cannot allow some expressions to be formed. For example, we cannot say, as in LTL: "All paths in which have p on them, also have q on them".

This expression can be translated in LTL as follows:

$$Fp \rightarrow Fq$$

That's all we need?

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

**Improving our
language**

Model checking
algorithms

References

We can try expressing it as $AFp \rightarrow AFq$ but it does not have the same meaning. This one statement means "If all paths have a p along them, then all paths have a q along then"

That's all we need?

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

We can try expressing it as $AFp \rightarrow AFq$ but it does not have the same meaning. This one statement means "If all paths have a p along them, then all paths have a q along then"

We can try to translate it as $AG(p \rightarrow AFq)$ which is closer, but not exactly the same. This one means "for all paths, in all states on the future, if they hold p then, all paths will eventually hold q "

Presenting CTL*

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

**Improving our
language**

Model checking
algorithms

References

For this, we can extend the CTL by dropping the constraint that every temporal operator (X , U , F , G) has to be associated with an unique path quantifier (A , E).

Presenting CTL*

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

For this, we can extend the CTL by dropping the constraint that every temporal operator (X , U , F , G) has to be associated with an unique path quantifier (A , E).

This allows us to generate some statements:

Presenting CTL*

Statements only possible with CTL*

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “In all possible paths, q is true until r is true or p is true until r is true”: $A[qUr \vee pUr]$

Presenting CTL*

Statements only possible with CTL*

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “In all possible paths, q is true until r is true or p is true until r is true”: $A[qUr \vee pUr]$
- “There is a path in which p eventually occurring will occur in all states”: $E[GFp]$

Presenting CTL*

Statements only possible with CTL*

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- “In all possible paths, q is true until r is true or p is true until r is true”: $A[qUr \vee pUr]$
- “There is a path in which p eventually occurring will occur in all states”: $E[GFp]$
- “In all paths, p will occur in the next state or in the next of the next”: $A[Xp \vee XXp]$

Presenting CTL*

CTL* syntax

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

The syntax of CTL* can be defined with the BNF bellow:

$$\phi ::= \perp \mid \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid A[\alpha] \mid E[\alpha] \mid$$

$$\alpha ::= \phi \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \alpha U \alpha \mid G\alpha \mid F\alpha \mid X\alpha \mid$$

With the same meanings of each operator.

Presenting CTL*

$LTL \subset CTL^*$ and $CTL \subset CTL^*$

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Although we don't define path operators to LTL we can assume that it consider in all paths. Therefore, we can say that a formula ϕ in LTL is a formula $A[\phi]$ in CTL^* ;

Presenting CTL*

$LTL \subset CTL^*$ and $CTL \subset CTL^*$

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Although we don't define path operators to LTL we can assume that it consider in all paths. Therefore, we can say that a formula ϕ in LTL is a formula $A[\phi]$ in CTL^* ;
For CTL, it is trivial;

The CTL model-checking algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

- Given a property of a program expressed in a temporal logic, the model checker checks if the states of the program satisfy the formula.

The CTL model-checking algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

- Given a property of a program expressed in a temporal logic, the model checker checks if the states of the program satisfy the formula.
- Obviously, we use CTL formulas.

The CTL model-checking algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Given a property of a program expressed in a temporal logic, the model checker checks if the states of the program satisfy the formula.
- Obviously, we use CTL formulas.
- There are two different ways of working with model checking:

The CTL model-checking algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Given a property of a program expressed in a temporal logic, the model checker checks if the states of the program satisfy the formula.
- Obviously, we use CTL formulas.
- There are two different ways of working with model checking:
Given a model, a state and a formula, tells if the model and the state satisfies the formula;

The CTL model-checking algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Given a property of a program expressed in a temporal logic, the model checker checks if the states of the program satisfy the formula.
- Obviously, we use CTL formulas.
- There are two different ways of working with model checking:
Given a model, a state and a formula, tells if the model and the state satisfies the formula;
Given a model and a formula, returns the set of states that satisfies the formula.

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

We present an algorithm that, given a model and a CTL formula, outputs the set of states of the model that satisfy the formula.

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

The algorithm deals explicitly only with some of the CTL connectives; for the others, it transforms them to their equivalent form in terms of the minimal set of connectives previously defined: $\{\perp, \neg, \wedge, AF, EU, EX\}$

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

Here is the algorithm:

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

Here is the algorithm:

INPUT: a CTL model $\mathcal{M} = (S, \rightarrow, L)$ and a CTL formula ϕ .

OUTPUT: the set of states of \mathcal{M} which satisfies ϕ .

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Here is the algorithm:

INPUT: a CTL model $\mathcal{M} = (S, \rightarrow, L)$ and a CTL formula ϕ .

OUTPUT: the set of states of \mathcal{M} which satisfies ϕ .

- First, rewrite ϕ in terms of $\perp, \neg, \wedge, AF, EU$ and EX .

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Here is the algorithm:

INPUT: a CTL model $\mathcal{M} = (S, \rightarrow, L)$ and a CTL formula ϕ .

OUTPUT: the set of states of \mathcal{M} which satisfies ϕ .

- First, rewrite ϕ in terms of $\perp, \neg, \wedge, AF, EU$ and EX .
- Next, label the states of \mathcal{M} with the subformulas of ϕ that are satisfied there, starting with the smallest subformulas and working outwards towards ϕ .

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

Suppose ψ is a subformula of ϕ and states satisfying all the immediate subformulas of ψ have already been labeled. We determine by a case analysis which states to label with ψ . If ψ is

- \perp : then no states are labeled with \perp .

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

Suppose ψ is a subformula of ϕ and states satisfying all the immediate subformulas of ψ have already been labeled. We determine by a case analysis which states to label with ψ . If ψ is

- \perp : then no states are labeled with \perp .
- p : then label s with p if $p \in L(s)$.

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

Suppose ψ is a subformula of ϕ and states satisfying all the immediate subformulas of ψ have already been labeled. We determine by a case analysis which states to label with ψ . If ψ is

- \perp : then no states are labeled with \perp .
- p : then label s with p if $p \in L(s)$.
- $\psi_1 \wedge \psi_2$: label s with $\psi_1 \wedge \psi_2$ if s is already labeled both with ψ_1 and with ψ_2 .

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Suppose ψ is a subformula of ϕ and states satisfying all the immediate subformulas of ψ have already been labeled. We determine by a case analysis which states to label with ψ . If ψ is

- \perp : then no states are labeled with \perp .
- p : then label s with p if $p \in L(s)$.
- $\psi_1 \wedge \psi_2$: label s with $\psi_1 \wedge \psi_2$ if s is already labeled both with ψ_1 and with ψ_2 .
- $\neg\psi_1$: label s with $\neg\psi_1$ if s is not already labeled with ψ_1 .

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silva

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Suppose ψ is a subformula of ϕ and states satisfying all the immediate subformulas of ψ have already been labeled. We determine by a case analysis which states to label with ψ . If ψ is

- \perp : then no states are labeled with \perp .
- p : then label s with p if $p \in L(s)$.
- $\psi_1 \wedge \psi_2$: label s with $\psi_1 \wedge \psi_2$ if s is already labeled both with ψ_1 and with ψ_2 .
- $\neg\psi_1$: label s with $\neg\psi_1$ if s is not already labeled with ψ_1 .
- $AF\psi_1$:
 - If any state s is labeled with ψ_1 , label it with $AF\psi_1$.
 - Repeat: label any state with $AF\psi_1$ if all successor states are labeled with $AF\psi_1$, until there is no change.

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Suppose ψ is a subformula of ϕ and states satisfying all the immediate subformulas of ψ have already been labeled. We determine by a case analysis which states to label with ψ . If ψ is

- $E[\psi_1 U \psi_2]$:
 - If any state s is labeled with ψ_2 , label it with $E[\psi_1 U \psi_2]$.
 - Repeat: label any state with $E[\psi_1 U \psi_2]$ if it is labeled with ϕ_1 and at least one of its successors is labeled with $E[\psi_1 U \psi_2]$, until there is no change.

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

Suppose ψ is a subformula of ϕ and states satisfying all the immediate subformulas of ψ have already been labeled. We determine by a case analysis which states to label with ψ . If ψ is

- $E[\psi_1 U \psi_2]$:
 - If any state s is labeled with ψ_2 , label it with $E[\psi_1 U \psi_2]$.
 - Repeat: label any state with $E[\psi_1 U \psi_2]$ if it is labeled with ψ_1 and at least one of its successors is labeled with $E[\psi_1 U \psi_2]$, until there is no change.
- $EX\psi_1$: label any state with $EX\psi_1$ if one of its successors is labeled with ψ_1 .

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

- Having performed the labeling for all the subformulas of ϕ (including ϕ itself), we output the states which are labeled ϕ .

The CTL model-checking algorithm

The labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- Having performed the labeling for all the subformulas of ϕ (including ϕ itself), we output the states which are labeled ϕ .
- The complexity of this algorithm is $O(fV(V + E))$, where f is the number of connectives in the formula, V is the number of states and E is the number of transitions.

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

- Here, we present a simple, pretty pseudocode for the labeling algorithm.

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

- Here, we present a simple, pretty pseudocode for the labeling algorithm.
- The program *SAT* expects a tree-structured CTL formula constructed by means of the BNF showed earlier.

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

function $SAT(\phi)$ **begin**

case

ϕ is \top : **return** S

ϕ is \perp : **return** \emptyset

ϕ is atomic: **return** $\{s \in S \mid \phi \in L(s)\}$

ϕ is $\neg\phi_1$: **return** $S - SAT(\phi_1)$

ϕ is $\phi_1 \wedge \phi_2$: **return** $SAT(\phi_1) \cap SAT(\phi_2)$

ϕ is $\phi_1 \vee \phi_2$: **return** $SAT(\phi_1) \cup SAT(\phi_2)$

ϕ is $\phi_1 \rightarrow \phi_2$: **return** $SAT(\neg\phi_1 \vee \phi_2)$

ϕ is $AX\phi_1$: **return** $SAT(\neg EX\neg\phi_1)$

ϕ is $EX\phi_1$: **return** $SAT_{EX}(\phi_1)$

ϕ is $A[\phi_1 U\phi_2]$: **return**

$SAT(\neg(E[\neg\phi_2 U(\neg\phi_1 \wedge \phi_2)] \vee EG\neg\phi_2))$

ϕ is $E[\phi_1 U\phi_2]$: **return** $SAT_{EU}(\phi_1, \phi_2)$

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

```
 $\phi$  is  $EF\phi_1$ : return  $SAT(E(\top U\phi_1))$   
 $\phi$  is  $EG\phi_1$ : return  $SAT(\neg AF\neg\phi_1)$   
 $\phi$  is  $AF\phi_1$ : return  $SAT_{AF}(\phi_1)$   
 $\phi$  is  $AG\phi_1$ : return  $SAT(\neg EF\neg\phi_1)$   
end case  
end function
```

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

- SAT handles with the easy cases (the propositional) directly and passes more complicated cases (the temporal) on to special procedures.

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

- SAT handles with the easy cases (the propositional) directly and passes more complicated cases (the temporal) on to special procedures.
- These special procedures uses the following functions:

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

- SAT handles with the easy cases (the propositional) directly and passes more complicated cases (the temporal) on to special procedures.
- These special procedures uses the following functions:
$$pre_{\exists}(Y) = \{s \in S \mid \exists s'(s \rightarrow s' \wedge s' \in Y)\}$$
$$pre_{\forall}(Y) = \{s \in S \mid \forall s'(s \rightarrow s' \longrightarrow s' \in Y)\}$$

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- SAT handles with the easy cases (the propositional) directly and passes more complicated cases (the temporal) on to special procedures.
- These special procedures uses the following functions:
$$pre_{\exists}(Y) = \{s \in S \mid \exists s'(s \rightarrow s' \wedge s' \in Y)\}$$
$$pre_{\forall}(Y) = \{s \in S \mid \forall s'(s \rightarrow s' \longrightarrow s' \in Y)\}$$
- 'pre' denotes travelling backwards along the transition relation.

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- SAT handles with the easy cases (the propositional) directly and passes more complicated cases (the temporal) on to special procedures.
- These special procedures uses the following functions:
$$pre_{\exists}(Y) = \{s \in S \mid \exists s'(s \rightarrow s' \wedge s' \in Y)\}$$
$$pre_{\forall}(Y) = \{s \in S \mid \forall s'(s \rightarrow s' \longrightarrow s' \in Y)\}$$
- 'pre' denotes travelling backwards along the transition relation.
- pre_{\exists} returns set of states of S which *can* make a transition into S .

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- SAT handles with the easy cases (the propositional) directly and passes more complicated cases (the temporal) on to special procedures.
- These special procedures uses the following functions:
$$pre_{\exists}(Y) = \{s \in S \mid \exists s'(s \rightarrow s' \wedge s' \in Y)\}$$
$$pre_{\forall}(Y) = \{s \in S \mid \forall s'(s \rightarrow s' \longrightarrow s' \in Y)\}$$
- 'pre' denotes travelling backwards along the transition relation.
- pre_{\exists} returns set of states of S which *can* make a transition into S .
- pre_{\forall} returns the set of states of S which make transitions *only* into Y .

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

**Model checking
algorithms**

References

The pseudocode for the special procedures of SAT are the following.

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

```
function  $SAT_{EX}(\phi)$   
local var  $X, Y$   
begin  
     $X := SAT(\phi)$   
     $Y := pre_{\exists}(X)$   
    return  $Y$   
end
```

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

```
function  $SAT_{AF}(\phi)$ 
local var  $X, Y$ 
begin
     $X := S$ 
     $Y := SAT(X)$ 
    repeat until  $X = Y$ 
    begin
         $X := Y$ 
         $Y := Y \cup pre_{\forall}(Y)$ 
    end
    return  $Y$ 
end
```

The CTL model-checking algorithm

A pseudocode for labelling algorithm

Computation
Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

```
function  $SAT_{EU}(\phi, \psi)$ 
local var  $W, X, Y$ 
begin
     $W := SAT(\phi)$ 
     $X := S$ 
     $Y := SAT(\psi)$ 
    repeat until  $X = Y$ 
    begin
         $X := Y$ 
         $Y := Y \cup (W \cap pre_{\exists}(Y))$ 
    end
    return  $Y$ 
end
```

References

Computation Tree Logic

Luis Tertulino &
Ronaldo Silveira

In previous
chapters...

Introduction

How to
communicate
Syntax of CTL
Semantics of
CTL

Some examples of
what we can say

More about
semantics
Equivalences

Improving our
language

Model checking
algorithms

References

- [1] Mordechai Ben-Ari. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.
- [2] Michael Huth and Mark Ryan. *Logic in Computer Science*. Cambridge University Press, 2004.
- [3] Mordechai Ben-Ari, Zohar Manna, and Amir Pnueli. The temporal logic of branching time. 1981.
- [4] Alessandro Artale. Formal methods lecture iv: Computation tree logic (ctl). URL <http://www.inf.unibz.it/~artale/FM/slide4.pdf>.