

Teoria współbieżności

Piotr Hofman

Theoretical aspects of concurrency

Lecture 3-4

HML and Bisimulation

Example

Consider a software controlling a car. We would like to have a property that in any state there is a possibility to brake.

Example

Consider a software controlling a car. We would like to have a property that in any state there is a possibility to brake.

- Question: How to formalise it in LTL?

Example

Consider a software controlling a car. We would like to have a property that in any state there is a possibility to brake.

- Question: How to formalise it in LTL?
- Can we express this in LTL?

Example

Consider a software controlling a car. We would like to have a property that in any state there is a possibility to brake.

- Question: How to formalise it in LTL?
- Can we express this in LTL?

Definition

We say that two objects are distinguished by a formula ϕ if ϕ is satisfied for one object and not satisfied by the second.

Example

Consider a software controlling a car. We would like to have a property that in any state there is a possibility to brake.

- Question: How to formalise it in LTL?
- Can we express this in LTL?

Definition

We say that two objects are distinguished by a formula ϕ if ϕ is satisfied for one object and not satisfied by the second.

Example

Consider a software controlling a car. We would like to have a property that in any state there is a possibility to brake.

- Question: How to formalise it in LTL?
- Can we express this in LTL?

Definition

We say that two objects are distinguished by a formula ϕ if ϕ is satisfied for one object and not satisfied by the second.

Example

Consider a software controlling a car. We would like to have a property that in any state there is a possibility to brake.

- Question: How to formalise it in LTL?
- Can we express this in LTL?

Definition

We say that two objects are distinguished by a formula ϕ if ϕ is satisfied for one object and not satisfied by the second.

Lemma

If two Kripke structures S and S' have the same traces i.e. $\text{TR}(S) = \text{TR}(S')$ then for any LTL formula ϕ does not distinguish S and S' .

Concluding: some interesting properties can not be analysed if we look only into traces.

Transition Tree (derivation tree)

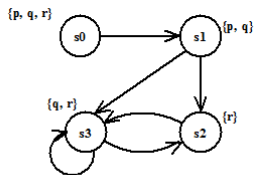


Figure 15. Example of a Kripke structure

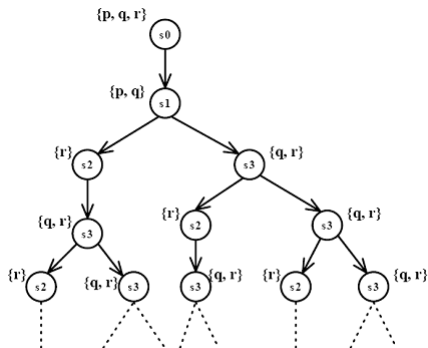


Figure 16. The unfolded tree of the Kripke structure of Figure 15

HMS Logic

Hennessy–Milner logic

A minimum to speak about tree, $\phi =$

- tt (*true*), ff (*false*),
- $\phi_1 \vee \phi_2$,
- $\phi_1 \wedge \phi_2$,
- $\Box\phi$ or $\square\phi$ or $AX\phi$,
- $\Diamond\phi$ or $\lozenge\phi$ or $EX\phi$.

Semantic of HML

Formula is always evaluated in a node of the transition tree.

- tt holds in every node.

Semantic of HML

Formula is always evaluated in a node of the transition tree.

- tt holds in every node.
- ff holds in the empty set.

Semantic of HML

Formula is always evaluated in a node of the transition tree.

- tt holds in every node.
- ff holds in the empty set.
- if ϕ_1 holds in the node s or ϕ_2 holds in the node s then $\phi_1 \vee \phi_2$ holds in s .

Semantic of HML

Formula is always evaluated in a node of the transition tree.

- tt holds in every node.
- ff holds in the empty set.
- if ϕ_1 holds in the node s or ϕ_2 holds in the node s then $\phi_1 \vee \phi_2$ holds in s .
- if ϕ_1 holds in the node s and ϕ_2 holds in the node s then $\phi_1 \wedge \phi_2$ holds in s .

Semantic of HML

Formula is always evaluated in a node of the transition tree.

- tt holds in every node.
- ff holds in the empty set.
- if ϕ_1 holds in the node s or ϕ_2 holds in the node s then $\phi_1 \vee \phi_2$ holds in s .
- if ϕ_1 holds in the node s and ϕ_2 holds in the node s then $\phi_1 \wedge \phi_2$ holds in s .
- $\langle \rangle \phi$ holds in s if there is s' a child of s such that ϕ holds in s' .

Semantic of HML

Formula is always evaluated in a node of the transition tree.

- tt holds in every node.
- ff holds in the empty set.
- if ϕ_1 holds in the node s or ϕ_2 holds in the node s then $\phi_1 \vee \phi_2$ holds in s .
- if ϕ_1 holds in the node s and ϕ_2 holds in the node s then $\phi_1 \wedge \phi_2$ holds in s .
- $\langle \rangle \phi$ holds in s if there is s' a child of s such that ϕ holds in s' .
- $[]\phi$ holds in s if for every s' a child of s we have that ϕ holds in s' .

Semantic of HML

Formula is always evaluated in a node of the transition tree.

- tt holds in every node.
- ff holds in the empty set.
- if ϕ_1 holds in the node s or ϕ_2 holds in the node s then $\phi_1 \vee \phi_2$ holds in s .
- if ϕ_1 holds in the node s and ϕ_2 holds in the node s then $\phi_1 \wedge \phi_2$ holds in s .
- $\langle \rangle \phi$ holds in s if there is s' a child of s such that ϕ holds in s' .
- $\Box \phi$ holds in s if for every s' a child of s we have that ϕ holds in s' .
- Later we will add to it modal operators to speak about descendants instead of children.

Semantic of HML

Formula is always evaluated in a node of the transition tree.

- tt holds in every node.
- ff holds in the empty set.
- if ϕ_1 holds in the node s or ϕ_2 holds in the node s then $\phi_1 \vee \phi_2$ holds in s .
- if ϕ_1 holds in the node s and ϕ_2 holds in the node s then $\phi_1 \wedge \phi_2$ holds in s .
- $\langle \rangle \phi$ holds in s if there is s' a child of s such that ϕ holds in s' .
- $\Box \phi$ holds in s if for every s' a child of s we have that ϕ holds in s' .
- Later we will add to it modal operators to speak about descendants instead of children.

Question what is the meaning of $\Box ff$?

What is the power of HML

Question: What are the trees that can be distinguished using HML?

What is the power of HML

Question: What are the trees that can be distinguished using HML?

- What can we distinguish by formulas with modal depth 1.

What is the power of HML

Question: What are the trees that can be distinguished using HML?

- What can we distinguish by formulas with modal depth 1.
- What can we distinguish by formulas with modal depth 2.

What is the power of HML

Question: What are the trees that can be distinguished using HML?

- What can we distinguish by formulas with modal depth 1.
- What can we distinguish by formulas with modal depth 2.
- What can we distinguish by formulas with modal depth 3.

What is the power of HML

Question: What are the trees that can be distinguished using HML?

- What can we distinguish by formulas with modal depth 1.
- What can we distinguish by formulas with modal depth 2.
- What can we distinguish by formulas with modal depth 3.
- What can we distinguish by formulas with modal depth 4.

What is the power of HML

Question: What are the trees that can be distinguished using HML?

- What can we distinguish by formulas with modal depth 1.
- What can we distinguish by formulas with modal depth 2.
- What can we distinguish by formulas with modal depth 3.
- What can we distinguish by formulas with modal depth 4.

Unlabelled case:

Definition

Bisimulation B is any relation on a set of configurations (nodes) that satisfies following conditions

- 1 if $(s, s') \in B$ then for every t such that $s \rightarrow t$ there is t' such that $s' \rightarrow t'$ and $(t, t') \in B$,
- 2 if $(s, s') \in B$ then for every t' such that $s' \rightarrow t'$ there is t such that $s \rightarrow t$ and $(t, t') \in B$.

We denote it by $s \sim_B s'$.

What is the power of HML

Question: What are the trees that can be distinguished using HML?

- What can we distinguish by formulas with modal depth 1.
- What can we distinguish by formulas with modal depth 2.
- What can we distinguish by formulas with modal depth 3.
- What can we distinguish by formulas with modal depth 4.

Labelled case:

Definition

Bisimulation B is any relation on a set of configurations (nodes) that satisfies following conditions

- 1 if $(s, s') \in B$ then $L(s) = L(s')$,
- 2 if $(s, s') \in B$ then for every t such that $s \rightarrow t$ there is t' such that $s' \rightarrow t'$ and $(t, t') \in B$,
- 3 if $(s, s') \in B$ then for every t' such that $s' \rightarrow t'$ there is t such that $s \rightarrow t$ and $(t, t') \in B$.

We denote it by $s \sim_B s'$.

Bisimulation relation

Theorem

A pair of configurations (s, s') can not be distinguished by HML formula if and only if there is a bisimulation relation B such that $s \sim_B s'$.

We need a few lemmas.

Bisimulation relation

Theorem

A pair of configurations (s, s') can not be distinguished by HML formula if and only if there is a bisimulation relation B such that $s \sim_B s'$.

We need a few lemmas.

Lemma

Union of bisimulations is a bisimulation.

Bisimulation relation

Theorem

A pair of configurations (s, s') can not be distinguished by HML formula if and only if there is a bisimulation relation B such that $s \sim_B s'$.

We need a few lemmas.

Lemma

Union of bisimulations is a bisimulation.

Corollary

There is a biggest bisimulation.

Bisimulation relation

Theorem

A pair of configurations (s, s') can not be distinguished by HML formula if and only if there is a bisimulation relation B such that $s \sim_B s'$.

We need a few lemmas.

Lemma

Union of bisimulations is a bisimulation.

Definition

- The biggest bisimulation is called the bisimilarity relation and denoted by \sim .
- We say that two configurations (s, s') are bisimilar if $s \sim s'$.

Bisimulation relation

Theorem

A pair of configurations (s, s') can not be distinguished by HML formula if and only if there is a bisimulation relation B such that $s \sim_B s'$.

We need a few lemmas.

Lemma

Union of bisimulations is a bisimulation.

Definition

- The biggest bisimulation is called the bisimilarity relation and denoted by \sim .
- We say that two configurations (s, s') are bisimilar if $s \sim s'$.
- Bisimilarity is an equivalence relation.

The proof of the theorem (idea).



- 1 By negation, suppose there is a formula ϕ that distinguishes (s, s') , we prove that (s, s') is not an element of any bisimulation relation.



The proof of the theorem (idea).



- 1 By negation, suppose there is a formula ϕ that distinguishes (s, s') , we prove that (s, s') is not an element of any bisimulation relation.
- 2 By induction on the modal depth of the formula.



The proof of the theorem (idea).



- 1 By negation, suppose there is a formula ϕ that distinguishes (s, s') , we prove that (s, s') is not an element of any bisimulation relation.
- 2 By induction on the modal depth of the formula.



- 1 Let X be a set of pairs of states that can not be distinguished by HML.
- 2 We prove that X is a bisimulation relation.

CTL and CTL*

An extension of HML - CTL

Definition

- 1 A state formula $\phi = tt | \neg\phi_1 | \phi_1 \wedge \phi_2 | p_i | A\alpha | E\alpha$
- 2 A path formula (restricted LTL) $\alpha = X\phi_1 | \phi_1 U \phi_2 | F\phi_1 | G\phi_1$
- 3 where ϕ are state formulas and α are path formulas.

Semantics

- p_i means that the predicate p_i holds in the configuration in which the formula is evaluated (current configuration).
- $A\alpha$ for every run r starting at the current configuration the formula α holds for a sequence of states of r .
- $E\alpha$ there is a run r starting at the current configuration such that the formula α holds for the sequence of states of r .

Exercise

- $F\phi = trueU\phi$,
- $G\phi = \neg(F\neg\phi)$

Which of the following pairs of CTL formulas are equivalent? For those which are not, find a model of one of the pair which is not a model of the other:^a

^aexercise from

<https://www.win.tue.nl/~andova/education/2IF25/Ex2Solutions.pdf>

Exercise

- $F\phi = \text{true}U\phi$,
- $G\phi = \neg(F\neg\phi)$

Which of the following pairs of CTL formulas are equivalent? For those which are not, find a model of one of the pair which is not a model of the other:^a

- 1 $EF\phi$ and $EG\phi$,

^aexercise from

<https://www.win.tue.nl/~andova/education/2IF25/Ex2Solutions.pdf>

Exercise

- $F\phi = trueU\phi$,
- $G\phi = \neg(F\neg\phi)$

Which of the following pairs of CTL formulas are equivalent? For those which are not, find a model of one of the pair which is not a model of the other:^a

- 1 $EF\phi$ and $EG\phi$,
- 2 $EF\phi \vee EF\tau$ and $EF(\phi \vee \tau)$,

^aexercise from

<https://www.win.tue.nl/~andova/education/2IF25/Ex2Solutions.pdf>

Exercise

- $F\phi = trueU\phi$,
- $G\phi = \neg(F\neg\phi)$

Which of the following pairs of CTL formulas are equivalent? For those which are not, find a model of one of the pair which is not a model of the other:^a

- 1 $EF\phi$ and $EG\phi$,
- 2 $EF\phi \vee EF\tau$ and $EF(\phi \vee \tau)$,
- 3 $AF\phi \vee AF\tau$ and $AF(\phi \vee \tau)$,

^aexercise from

<https://www.win.tue.nl/~andova/education/2IF25/Ex2Solutions.pdf>

Exercise

- $F\phi = trueU\phi$,
- $G\phi = \neg(F\neg\phi)$

Which of the following pairs of CTL formulas are equivalent? For those which are not, find a model of one of the pair which is not a model of the other:^a

- 1 $EF\phi$ and $EG\phi$,
- 2 $EF\phi \vee EF\tau$ and $EF(\phi \vee \tau)$,
- 3 $AF\phi \vee AF\tau$ and $AF(\phi \vee \tau)$,
- 4 $AF\neg\phi$ and $\neg EG\phi$.

^aexercise from

<https://www.win.tue.nl/~andova/education/2IF25/Ex2Solutions.pdf>

Exercise

- $F\phi = trueU\phi$,
- $G\phi = \neg(F\neg\phi)$

Which of the following pairs of CTL formulas are equivalent? For those which are not, find a model of one of the pair which is not a model of the other:^a

- 1 $EF\phi$ and $EG\phi$,
 - 2 $EF\phi \vee EF\tau$ and $EF(\phi \vee \tau)$,
 - 3 $AF\phi \vee AF\tau$ and $AF(\phi \vee \tau)$,
 - 4 $AF\neg\phi$ and $\neg EG\phi$.
- Write a CTL formula which stays that there is always a possibility of braking.

^aexercise from

<https://www.win.tue.nl/~andova/education/2IF25/Ex2Solutions.pdf>

LTL vs. CTL

Evaluation of LTL in a state.

We consider all traces starting in a given state.

LTL vs. CTL

Evaluation of LTL in a state.

We consider all traces starting in a given state.

Lemma

There are properties which can be expressed in LTL and can not in CTL and vice versa.

LTL vs. CTL

Evaluation of LTL in a state.

We consider all traces starting in a given state.

Lemma

There are properties which can be expressed in LTL and can not in CTL and vice versa.

Example (CTL not in LTL)

$A\ G\ E\ F\ (\text{brake} == \text{true})$

LTL vs. CTL

Evaluation of LTL in a state.

We consider all traces starting in a given state.

Lemma

There are properties which can be expressed in LTL and can not in CTL and vice versa.

Example (CTL not in LTL)

$A \ G \ E \ F \text{ (brake} == \text{true)}$

Two systems have the same sets of traces but only one satisfies the formula in CTL.

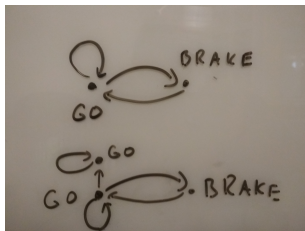
LTL vs. CTL

Evaluation of LTL in a state.

We consider all traces starting in a given state.

Lemma

There are properties which can be expressed in LTL and can not in CTL and vice versa.



Example (CTL not in LTL)

$A \ G \ E \ F \text{ (brake} == \text{true)}$

Two systems have the same sets of traces but only one satisfies the formula in CTL.

LTL vs. CTL

Evaluation of LTL in a state.

We consider all traces starting in a given state.

Lemma

There are properties which can be expressed in LTL and can not in CTL and vice versa.

Example (LTL not in CTL)

A F G (black = true)

For all runs there will be a moment from which onward holds (black = true).

LTL vs. CTL

Evaluation of LTL in a state.

We consider all traces starting in a given state.

Lemma

There are properties which can be expressed in LTL and can not in CTL and vice versa.

Example (LTL not in CTL)

A F G (black = true)

For all runs there will be a moment from which onward holds (black = true). The idea is to construct two sequences of systems T_i and T'_i such that:

- 1 $(T_i, s_0) \models AFG(\text{black} = \text{true})$ but $(T'_i, s_0) \not\models AFG(\text{black} = \text{true})$.
- 2 (T_i, s_0) and (T'_i, s'_0) are not distinguished by any CTL formula of the modal depth i .

LTL vs. CTL

Evaluation of LTL in a state.

We consider all traces starting in a given state.

Lemma




There are properties which can be expressed in LTL and can not in CTL and vice versa.

Example (LTL not in CTL)

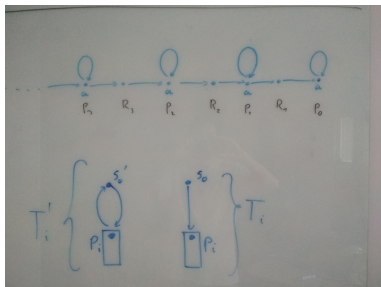
A F G (black = true)

For all runs there will be a moment from which onward holds (black = true). The idea is to construct two sequences of systems T_i and T'_i such that:

- 1 $(T_i, s_0) \models AFG(\text{black} = \text{true})$ but $(T'_i, s_0) \not\models AFG(\text{black} = \text{true})$.
- 2 (T_i, s_0) and (T'_i, s'_0) are not distinguished by any CTL formula of the modal depth i .

Look: <https://www.youtube.com/watch?v=0Af7q3X71-o> (min 58)   

Proof.



Let \sim_i not distinguishable by
CTL formulas of depth i .

Lemma (Auxiliary)

$$P_i \sim_i P_j \text{ for } i \leq j. \quad R_i \sim_i R_j \text{ for } i \leq j.$$

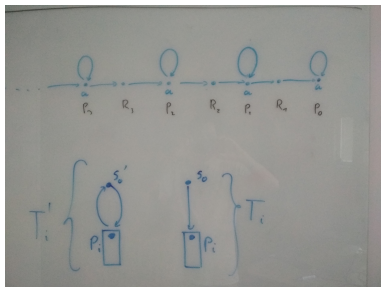
Proof.

Via induction on i , (the size of the formula).

Induction hypothesis:

$$P_k \sim_i P_j, R_k \sim_i R_j \text{ for } i \leq k \leq j.$$


Proof.



Let \sim_i not distinguishable by CTL formulas of depth i .

$$T_i \sim_i T'_i.$$

Via induction on i , (the size of the formula).

Induction hypothesis:

$$T_k \sim_i T'_j \text{ for } i \leq k \leq j.$$



Evaluation of CTL

Lemma

A CTL formula ϕ can be evaluated in time proportional to the length of the formula times size of the Kripke structure.

Proof.

By induction on the derivation tree of the formula. □

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' can not be distinguished by any CTL formula ϕ .

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' can not be distinguished by any CTL formula ϕ .

Proof \leftarrow (not distinguishable by CTL \implies bisimilar).

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' can not be distinguished by any CTL formula ϕ .

Proof \leftarrow (not distinguishable by CTL \implies bisimilar).

- If they are not distinguishable by CTL, then they are not distinguishable by HML with predicates.

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' can not be distinguished by any CTL formula ϕ .

Proof \leftarrow (not distinguishable by CTL \implies bisimilar).

- If they are not distinguishable by CTL, then they are not distinguishable by HML with predicates.
- Indeed, HML with predicates is a fragment of CTL.

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' can not be distinguished by any CTL formula ϕ .

Proof \leftarrow (not distinguishable by CTL \implies bisimilar).

- If they are not distinguishable by CTL, then they are not distinguishable by HML with predicates.
- Indeed, HML with predicates is a fragment of CTL.
- We already proved that if (s, s') are not distinguished by HML then they are bisimilar.

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' can not be distinguished by any CTL formula ϕ .

Proof \leftarrow (not distinguishable by CTL \implies bisimilar).

- If they are not distinguishable by CTL, then they are not distinguishable by HML with predicates.
- Indeed, HML with predicates is a fragment of CTL.
- We already proved that if (s, s') are not distinguished by HML then they are bisimilar.
- The proof for HML extended with predicates is the same.

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' can not be distinguished by any CTL formula ϕ .

Proof \leftarrow (not distinguishable by CTL \implies bisimilar).

- If they are not distinguishable by CTL, then they are not distinguishable by HML with predicates.
- Indeed, HML with predicates is a fragment of CTL.
- We already proved that if (s, s') are not distinguished by HML then they are bisimilar.
- The proof for HML extended with predicates is the same.
- The implication is proven.

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' can not be distinguished by any CTL formula ϕ .

Proof \leftarrow (not distinguishable by CTL \implies bisimilar).

- If they are not distinguishable by CTL, then they are not distinguishable by HML with predicates.
- Indeed, HML with predicates is a fragment of CTL.
- We already proved that if (s, s') are not distinguished by HML then they are bisimilar.
- The proof for HML extended with predicates is the same.
- The implication is proven.

Proof \rightarrow

We need to extend our understanding of bisimulation first.

Game characterisation of Bisimilarity

Definition

A bisimulation game is played in rounds between two players Spoiler and Duplicator. Arena is a set of pairs of configurations of the Kripke structure. Suppose that current pair of configurations is (s, s') .

Rules of a round are as follows:

- First Spoiler chooses one of configurations s or s' . Without loss of generality we may assume that it is s .
- Next he chooses a configuration t such that $s \rightarrow t$.
- Next Duplicator chooses a configuration t' such that $s' \rightarrow t'$ where s' is a configuration not chosen by Spoiler.
- The next round of the game will be played from (t, t') .

Winning conditions:

- If $L(s) \neq L(s')$ then Spoiler wins.
- If any player cannot make his part of the move then he loses.
- Infinite plays are won by Duplicator.

Lemma

Duplicator has a winning strategy in the bisimulation game starting from a pair of configurations (s, s') if and only if $s \sim s'$.

Lemma

Duplicator has a winning strategy in the bisimulation game starting from a pair of configurations (s, s') if and only if $s \sim s'$.

Lemma

A winning strategy for Spoiler is a tree.

Bisimulation and CTL

Lemma

Two configurations s and s' are bisimilar if and only if s and s' are not distinguished by any CTL formula ϕ .

Proof \rightarrow (bisimilar \implies not distinguishable by any CTL formula).

- If they are distinguishable by CTL, then they are distinguishable by some formula ϕ .
- We construct a winning strategy for Spoiler via induction on the modal depth of the formula.

Extend even more - CTL^*

Definition

A state formula $\phi = tt | \neg\phi_1 | \phi_1 \wedge \phi_2 | p_i | A\alpha | F\alpha$

A path formula (restricted LTL) $\alpha = \phi | \neg\alpha_1 | \alpha_1 \wedge \alpha_2 | X\alpha_1 | \alpha_1 U \alpha_2$

Semantics

- $A\alpha$ for every run r starting at the current configuration the formula α holds for $\mathbb{TR}(r)$.
- $F\alpha$ there is a run r starting at the current configuration such that the formula α holds for $\mathbb{TR}(r)$.

Fact

CTL^* subsumes CTL and LTL.

Bibliography

CTL, CTL*

<https://www.youtube.com/channel/UCUXDMaaobC01He1HBiFZnPQ>

Unites: 9, 10, 11.

Bisimulation + CTL and more:

[https://pdfs.semanticscholar.org/cb9f/
325389bd6ee5894dcf435159d34f9e20da2d.pdf](https://pdfs.semanticscholar.org/cb9f/325389bd6ee5894dcf435159d34f9e20da2d.pdf)