Teoria współbieżności

Piotr Hofman
Theoretical aspects of concurrency

- My email: piotrek.hofman@gmail.com
- My office: 4580

Outline

1. How to specify properties of a system?
    - LTL.
    - CTL.
    - Bisimulation.
2. How to model a system?
    - Process algebra.
    - Petri nets.

Assessment methods and assessment criteria

Oral exam 0 up to 15 point

- 3 questions each for 0-5 points

In the end of the semester I will provide a list of questions that may appear on the exam.

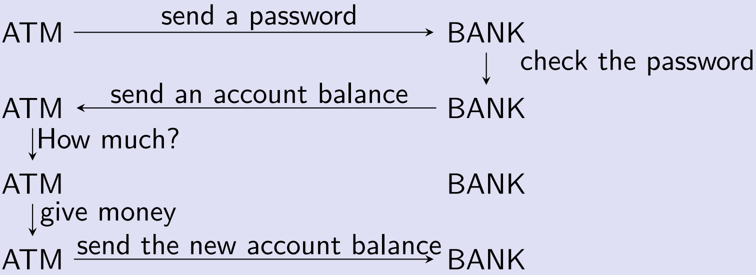
- $[0 - 8) \leftrightarrow 2$
- $[8 - 10) \leftrightarrow 3$
- $[10 - 11.5) \leftrightarrow 3+$
- $[11.5 - 13) \leftrightarrow 4$
- $[13 - 14) \leftrightarrow 4+$
- $[14 - 15] \leftrightarrow 5$

Basic problems with concurrent programs

Three basic threats in concurrent programming

Data corruption

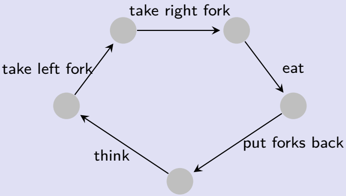
Consider a bank, an ATM, and a following protocol for withdrawing money:

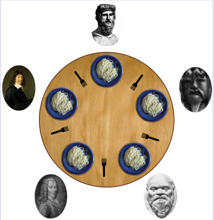


Three basic threats in concurrent programming

Deadlock

Consider philosophers working according a following schema:

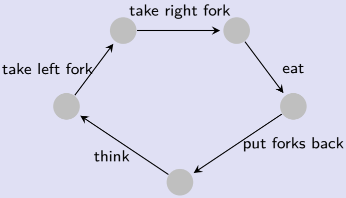


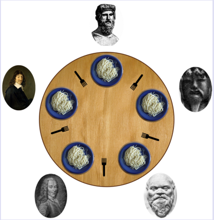


Three basic threats in concurrent programming

Deadlock

Consider philosophers working according a following schema:





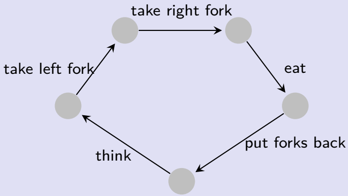
Solution

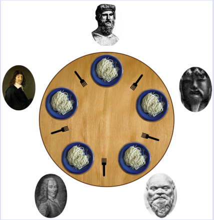
Priorities.

Three basic threats in concurrent programming

Starvation

Consider philosophers working according a following schema:





Solution

Priorities.

Kripke structures

Let $\mathbb{AP}$ be a set of atomic propositions, i.e. boolean expressions over variables, constants and predicate symbols.

Kripke structures

Let $\mathbb{AP}$ be a set of atomic propositions, i.e. boolean expressions over variables, constants and predicate symbols.

Definition

A Kripke structure over $\mathbb{AP}$ is a 4-tuple $M = (S, I, R, L)$ consisting of:

1. a finite set of states S,
2. a set of initial states $I \subseteq S$,
3. a transition relation $R \subseteq S \times S$ such that R is left-total, i.e., $\forall_{s \in S} \exists_{s' \in S}$ such that $(s, s') \in R$,
4. a labeling (or interpretation) function $L : S \to 2^{\mathbb{AP}}$.

Kripke structures

Let $\mathbb{AP}$ be a set of atomic propositions, i.e. boolean expressions over variables, constants and predicate symbols.

Definition

A Kripke structure over $\mathbb{AP}$ is a 4-tuple $M = (S, I, R, L)$ consisting of:

1. a finite set of states S,
2. a set of initial states $I \subseteq S$,
3. a transition relation $R \subseteq S \times S$ such that R is left-total, i.e., $\forall_{s \in S} \exists_{s' \in S}$ such that $(s, s') \in R$,
4. a labeling (or interpretation) function $L : S \to 2^{\mathbb{AP}}$.

Definition

By a *run* we mean a sequence of states interleaved with transitions, $s_1, t_1, s_2, t_2, s_3 \ldots$ such that $(s_i, s_{i+1}) = t_i$.

Runs

Definition

For a given run we define a *trace* as follows:

$$\mathbb{TR}(s_1, t_1, s_2, t_2, s_3 \ldots) = L(s_1), L(s_2), L(s_3) \ldots .$$

A set of traces of all possible infinite runs starting in I (one of initial states) of a given Kripke structure S is called *Traces* of S. We denoted it $\mathbb{TR}(S)$.

Runs

The first concept:
We describe properties of system by describing properties of the set of its traces.

Runs

The first concept:
We describe properties of system by describing properties of the set of its traces.

- Suppose that S is a Kripke structure that is a model of a given system.

Runs

The first concept:
We describe properties of system by describing properties of the set of its traces.

- Suppose that S is a Kripke structure that is a model of a given system.
- Let $\mathbb{X}$ be a set of infinite words that are witnesses of an error, say some possible memory corruption.

Runs

The first concept:
We describe properties of system by describing properties of the set of its traces.

- Suppose that S is a Kripke structure that is a model of a given system.
- Let $\mathbb{X}$ be a set of infinite words that are witnesses of an error, say some possible memory corruption.
- If $\mathbb{X} \cap \mathbb{TR}(S) = \emptyset$ then we know that the system does not allow for data corruption.

Runs

The first concept:
We describe properties of system by describing properties of the set of its traces.

- Suppose that S is a Kripke structure that is a model of a given system.
- Let $\mathbb{X}$ be a set of infinite words that are witnesses of an error, say some possible memory corruption.
- If $\mathbb{X} \cap \mathbb{TR}(S) = \emptyset$ then we know that the system does not allow for data corruption.
- Almost! It is under the assumption that model is correct and precise enough.

Runs

We may also partially specify a system by defining a set of correct behaviours.

- Suppose that S is a Kripke structure that is a model of a given system.
- Let $\mathbb{X}$ be a set of infinite words that are correct behaviours.

Runs

We may also partially specify a system by defining a set of correct behaviours.

- Suppose that S is a Kripke structure that is a model of a given system.
- Let $\mathbb{X}$ be a set of infinite words that are correct behaviours.
- If $\mathbb{X} \supseteq \mathbb{TR}(S)$ then we know that the system does not allow data corruption.

Exercises

$5 \to 3$ philosophers

- What are the predicates?
- How the Kripke structure looks like?
- What are the properties that Traces should satisfy for 5 philosophers?

Exercises

$5 \rightarrow 3$ philosophers

- What are the predicates?
- How the Kripke structure looks like?
- What are the properties that Traces should satisfy for 5 philosophers?

- Philosopher may eat only if he has two forks.
- Every philosopher eat infinite number of times.
- Rene Descartes eat and think infinite number of times.

Exercises

$5 \rightarrow 3$ philosophers

- What are the predicates?
- How the Kripke structure looks like?
- What are the properties that Traces should satisfy for 5 philosophers?

- Philosopher may eat only if he has two forks.
- Every philosopher eat infinite number of times.
- Rene Descartes eat and think infinite number of times.

How to specify the above properties?

Automaton

Traces are languages, so we can try specify properties with automata.
Let Σ be a set of letters (a finite alphabet).

Definition

Automaton is an ordered 5-tuple $A = (S, I, F, R, L)$ where:

1. S is a finite set of states,
2. I is a set of initial states, $I \subseteq S$,
3. F is a set of accepting states, $F \subseteq S$,
4. R is a transition relation $R \subseteq S \times S$,
5. L is a labelling (or interpretation) function $L : R \to \Sigma$.

Automaton

Traces are languages, so we can try specify properties with automata. Let Σ be a set of letters (a finite alphabet).

Definition

Automaton is an ordered 5-tuple $A = (S, I, F, R, L)$ where:

1. S is a finite set of states,
2. I is a set of initial states, $I \subseteq S$,
3. F is a set of accepting states, $F \subseteq S$,
4. R is a transition relation $R \subseteq S \times S$,
5. L is a labelling (or interpretation) function $L : R \to \Sigma$.

Definition

A language of an automaton A is a set of words $\subseteq \Sigma^*$ such that they can be read along the paths from an initial state to a final state.

Automaton

Traces are languages, so we can try specify properties with automata. Let Σ be a set of letters (a finite alphabet).

Definition

Automaton is an ordered 5-tuple $A = (S, I, F, R, L)$ where:

1. S is a finite set of states,
2. I is a set of initial states, $I \subseteq S$,
3. F is a set of accepting states, $F \subseteq S$,
4. R is a transition relation $R \subseteq S \times S$,
5. L is a labelling (or interpretation) function $L : R \to \Sigma$.

Problem

Traces are infinite words and words accepted by a non-deterministic automaton are finite.

Characterisations of languages of infinite words

Let's try to define automata on infinite words.

Characterisations of languages of infinite words

Let's try to define automata on infinite words.

- Attempt 1: all words with a prefix from a regular language.

Characterisations of languages of infinite words

Let's try to define automata on infinite words.

- Attempt 1: all words with a prefix from a regular language.
- Attempt 2: an infinite word is accepted if from some moment a run stays only in accepting states.

Characterisations of languages of infinite words

Let's try to define automata on infinite words.

- Attempt 1: all words with a prefix from a regular language.
- Attempt 2: an infinite word is accepted if from some moment a run stays only in accepting states.
- Attempt 3: Buchi automaton, word is accepted if it visits accepting states infinitely often.

Characterisations of languages of infinite words

Let's try to define automata on infinite words.

- Attempt 1: all words with a prefix from a regular language.
- Attempt 2: an infinite word is accepted if from some moment a run stays only in accepting states.
- Attempt 3: Buchi automaton, word is accepted if it visits accepting states infinitely often.
- Attempt 4: A generalised Buchi automaton.

Definition

A generalised Büchi automaton is an ordered 5-tuple $A = (S, I, F, R, L)$ where:

1. S is a finite set of states,
2. I is a set of initial states, $I \subseteq S$,
3. F is a finite set of sets $\{F_1 \ldots F_k\}$ of accepting states, $F_i \subseteq S$,
4. R is a transition relation $R \subseteq S \times S$,
5. L is a labelling (or interpretation) function $L : R \to \Sigma$.

Characterisations of languages of infinite words

Let's try to define automata on infinite words.

- Attempt 1: all words with a prefix from a regular language.
- Attempt 2: an infinite word is accepted if from some moment a run stays only in accepting states.
- Attempt 3: Buchi automaton, word is accepted if it visits accepting states infinitely often.
- Attempt 4: A generalised Buchi automaton.

Definition

A generalised Büchi automaton is an ordered 5-tuple $A = (S, I, F, R, L)$ where:

1. F is a finite set of sets $\{F_1 \ldots F_k\}$ of accepting states, $F_i \subseteq S$,

Characterisations of languages of infinite words

Let's try to define automata on infinite words.

- Attempt 1: all words with a prefix from a regular language.
- Attempt 2: an infinite word is accepted if from some moment a run stays only in accepting states.
- Attempt 3: Buchi automaton, word is accepted if it visits accepting states infinitely often.
- Attempt 4: A generalised Buchi automaton.

Definition

A generalised Büchi automaton is an ordered 5-tuple $A = (S, I, F, R, L)$ where:

1. F is a finite set of sets $\{F_1 \ldots F_k\}$ of accepting states, $F_i \subseteq S$,

Definition

A word is accepted if it visits infinitely often states in F_i for every $0 < i \leq k$.

An exercise

$\Sigma = \{a, b\}$.

An exercise

$\Sigma = \{a, b\}$.

1. Infinite number of a.

An exercise

$\Sigma = \{a, b\}$.

1. Infinite number of a.
2. Finite number of a.

An exercise

$\Sigma = \{a, b\}$.

1. Infinite number of a.
2. Finite number of a.
3. Number of b is infinite and number of a between every two b is divisible by 3.

An exercise

$\Sigma = \{a, b\}$.

1. Infinite number of a.
2. Finite number of a.
3. Number of b is infinite and number of a between every two b is divisible by 3.
4. Number of a between every two b is divisible by 3.

An exercise

$\Sigma = \{a, b\}$.

1. Infinite number of a.
2. Finite number of a.
3. Number of b is infinite and number of a between every two b is divisible by 3.
4. Number of a between every two b is divisible by 3.
5. Number of b is infinite and only finitely many times the number of a between every two consecutive b is divisible by 3.

An exercise

$\Sigma = \{a, b\}$.

1. Infinite number of a.
2. Finite number of a.
3. Number of b is infinite and number of a between every two b is divisible by 3.
4. Number of a between every two b is divisible by 3.
5. Number of b is infinite and only finitely many times the number of a between every two consecutive b is divisible by 3.
6. Number of b is finite and number of a between every two consecutive b is divisible by 3.

An exercise

$\Sigma = \{a, b\}$.

1. Infinite number of a.
2. Finite number of a.
3. Number of b is infinite and number of a between every two b is divisible by 3.
4. Number of a between every two b is divisible by 3.
5. Number of b is infinite and only finitely many times the number of a between every two consecutive b is divisible by 3.
6. Number of b is finite and number of a between every two consecutive b is divisible by 3.
7. Number of b is finite and number of a between any two b is not divisible by 3.

Büchi languages

Lemma

Languages recognised by Büchi automata and generalised Büchi automata are the same.

Büchi languages

Lemma

Languages recognised by Büchi automata and generalised Büchi automata are the same.

Lemma

The emptiness problem for Büchi automaton is NL complete.

Büchi languages

Lemma

Languages recognised by Büchi automata and generalised Büchi automata are the same.

Lemma

The emptiness problem for Büchi automaton is NL complete.

Lemma

Büchi languages are closed under:

1. union,
2. intersection,
3. complement (We will not do this)
4. determinisation does not work (we need to extend the model).

Kripke vs Büchi

Question

How to test if a system given via a Kripke structure satisfies a property given by a Büchi automaton?

Kripke vs Büchi

Question

How to test if a system given via a Kripke structure satisfies a property given by a Büchi automaton?

Lemma

For a given Kripke structure S there is a Büchi automaton A such that:

$$\mathbb{L}(A) = \mathbb{TR}(S).$$

Kripke vs Büchi

Question

How to test if a system given via a Kripke structure satisfies a property given by a Büchi automaton?

Lemma

For a given Kripke structure S there is a Büchi automaton A such that:

$$\mathbb{L}(A) = \mathbb{TR}(S).$$

So we can use intersection and test for non-emptiness.

(LTL) Linear temporal logic.

Automata are not very convenient to write, it would be better to have some query language.

Automata are not very convenient to write, it would be better to have some query language.

What are good properties of a query language?

1. It should be closed under Boolean operations.

Automata are not very convenient to write, it would be better to have some query language.

What are good properties of a query language?

1. It should be closed under Boolean operations.
2. It would be good to have a possibility to say that x is an immediate consequence of y.

Automata are not very convenient to write, it would be better to have some query language.

What are good properties of a query language?

1. It should be closed under Boolean operations.
2. It would be good to have a possibility to say that x is an immediate consequence of y.
3. It should allow to say something will happen eventually.

Automata are not very convenient to write, it would be better to have some query language.

What are good properties of a query language?

1. It should be closed under Boolean operations.
2. It would be good to have a possibility to say that x is an immediate consequence of y.
3. It should allow to say something will happen eventually.
4. It should allow to say something is always satisfied.

Automata are not very convenient to write, it would be better to have some query language.

What are good properties of a query language?

1. It should be closed under Boolean operations.
2. It would be good to have a possibility to say that x is an immediate consequence of y.
3. It should allow to say something will happen eventually.
4. It should allow to say something is always satisfied.
5. It should allow to say that if something is happening then later something different must happen.

Automata are not very convenient to write, it would be better to have some query language.

What are good properties of a query language?

1. It should be closed under Boolean operations.
2. It would be good to have a possibility to say that x is an immediate consequence of y.
3. It should allow to say something will happen eventually.
4. It should allow to say something is always satisfied.
5. It should allow to say that if something is happening then later something different must happen.
6. It should allow to say that something is happening infinitely often.

Automata are not very convenient to write, it would be better to have some query language.

What are good properties of a query language?

1. It should be closed under Boolean operations.
2. It would be good to have a possibility to say that x is an immediate consequence of y.
3. It should allow to say something will happen eventually.
4. It should allow to say something is always satisfied.
5. It should allow to say that if something is happening then later something different must happen.
6. It should allow to say that something is happening infinitely often.
7. It should allow to say that before something happens another thing holds.

Automata are not very convenient to write, it would be better to have some query language.

What are good properties of a query language?

1. It should be closed under Boolean operations.
2. It would be good to have a possibility to say that x is an immediate consequence of y.
3. It should allow to say something will happen eventually.
4. It should allow to say something is always satisfied.
5. It should allow to say that if something is happening then later something different must happen.
6. It should allow to say that something is happening infinitely often.
7. It should allow to say that before something happens another thing holds.
8. It should allow to say that some property holds after something happens.

Automata are not very convenient to write, it would be better to have some query language.

Definition

An *LTL* formula ϕ is generated according a following rules:

$$\phi \rightarrow true | p_i \in \mathbb{AP} | \phi_1 \wedge \phi_2 | \neg \phi_1 | X \phi_1 | \phi_1 U \phi_2$$

Semantics of LTL

$p_1 \rightarrow p_2 \rightarrow p_1 p_2 \rightarrow p_2 \rightarrow p_3 \rightarrow p_2 \rightarrow p_1 p_3 \rightarrow \cdots$

- $true \iff true$.
- $p_1 \iff p_1$ holds at position 0.
- $p_2 \iff p_2$ holds at position 0.
- $p_1 \wedge p_2 \iff p_1$ and p_2 holds at position 0.
- $\neg p_2 \iff p_2$ does not hold at position 0.
- $Xp_2 \iff p_2$ holds at position 1.
- $XX(p_1 \wedge p_2) \iff p_2$ and p_1 holds at position 2.
- $\neg(\neg p_1 \wedge \neg p_2) U p_3$
  $\iff$ there is $0 \leq j$ such that p_3 holds at position j and $\neg(\neg p_1 \wedge \neg p_2)$ holds for all $0 \leq i < j$.

Exercise

How to express:

1. Finally there will be a state in which p_2 holds.

Exercise

How to express:

1. Finally there will be a state in which p_2 holds.
   Fp_2

Exercise

How to express:

1. Finally there will be a state in which p_2 holds.
   Fp_2
2. Every state along the path satisfy p_2.

Exercise

How to express:

1. Finally there will be a state in which p_2 holds.
   Fp_2
2. Every state along the path satisfy p_2.
   Gp_2

Exercise

How to express:

1. Finally there will be a state in which p_2 holds.
   Fp_2
2. Every state along the path satisfy p_2.
   Gp_2
3. If p_2 holds at the state with index 2 then $p_3 \vee p_1$ holds in the state with index 4.

Exercise

How to express:

1. Finally there will be a state in which p_2 holds.
   Fp_2
2. Every state along the path satisfy p_2.
   Gp_2
3. If p_2 holds at the state with index 2 then $p_3 \vee p_1$ holds in the state with index 4.
4. If in some state p_1 is satisfied then in the future p_2 has to be satisfied.

How to verify LTL formula?

Lemma

Let *words*(ϕ) denotes a set of words that satisfy the LTL formula ϕ. For a given LTL formula ϕ one can construct an exponential size Büchi automaton B recognising exactly the same set of words, i.e.

$$\mathbb{L}(B) = \mathit{words}(\phi)$$

How to verify LTL formula?

Lemma

Let $\mathit{words}(\phi)$ denotes a set of words that satisfy the LTL formula ϕ. For a given LTL formula ϕ one can construct an exponential size Büchi automaton B recognising exactly the same set of words, i.e.

$$\mathbb{L}(B) = \mathit{words}(\phi)$$

1. Build an automaton A for the Kripke structure.
2. Build an automaton B for ϕ an LTL formula,
   or build an automaton B for $\neg\phi$.
3. Check non-emptiness of $\mathbb{L}(A) \cap \mathbb{L}(B)$.

Bibliography

Units from 3 to 8 from (ordered by date)
https://www.youtube.com/channel/UCUXDMaaobC01He1HBiFZnPQ/videos
There are a lot of videos first one is "A problem in concurrency".