Teoria współbieżności

Piotr Hofman

Theoretical aspects of concurrency

Lecture 5 — 6

How to compute the bisimilarity relation?

Properties of Bisimulation

Approximants

- Let B_0 be a set of all pairs of configurations.
- $(s, s') \in B_{i+1}$ if and only if:
  1. $L(s) = L(s')$,
  2. For all t such that $s \to t$ there is a $s' \to t'$ where $(t, t') \in B_i$.
  3. For all t' such that $s' \to t'$ there is a $s \to t$ where $(t, t') \in B_i$.

Properties of Bisimulation

Approximants

- Let B_0 be a set of all pairs of configurations.
- $(s, s') \in B_{i+1}$ if and only if:
  1. $L(s) = L(s')$,
  2. For all t such that $s \to t$ there is a $s' \to t'$ where $(t, t') \in B_i$.
  3. For all t' such that $s' \to t'$ there is a $s \to t$ where $(t, t') \in B_i$.

Lemma

The bisimilarity relation is the biggest fix point of approximants i.e. if $B_i = B_{i+1}$ *then* $B_i = \sim$.

Proof

$\subseteq$

1. We prove that B_{i+1} is a bisimulation relation (from the definition).
2. Take any pair $(s, s') \in B_{i+1}$. $s \to t$ and $s' \to t'$ such that $(t, t') \in B_i$. But then $(t, t') \in B_{i+1}$ so B_{i+1} is a bisimulation.

$\supseteq$

1. We prove that if $(s, s') \notin B_{i+1}$ then $(s, s') \notin \sim$.
2. We construct a winning strategy for Spoiler, by induction on i.
3. This works under the assumption that B_i converge for $i \leq \omega$ (an ordinal number).
4. This is a valid assumption for system with finite branching.

Approximation of the bisimilarity relation

Approximants

- Let B_0 be a set of all pairs of configurations.
- $(s, s') \in B_{i+1}$ if and only if:
  1. $L(s) = L(s')$,
  2. For all t such that $s \to t$ there is a $s' \to t'$ where $(t, t') \in B_i$.
  3. For all t' such that $s' \to t'$ there is a $s \to t$ where $(t, t') \in B_i$.

Question?

What if the starting point B_0 is different but it satisfies $B_0 \supseteq \sim$.

Approximation of the bisimilarity relation

Approximants

- Let B_0 be a set of all pairs of configurations.
- $(s, s') \in B_{i+1}$ if and only if:
  1. $L(s) = L(s')$,
  2. For all t such that $s \to t$ there is a $s' \to t'$ where $(t, t') \in B_i$.
  3. For all t' such that $s' \to t'$ there is a $s \to t$ where $(t, t') \in B_i$.

Question?

What if the starting point B_0 is different but it satisfies $B_0 \supseteq \sim$.

Lemma

Approximants are monotone i.e. if $X_i \subset Y_j$ then $X_{i+1} \subseteq Y_{j+1}$.

Algorithm 1

Lemma

The bisimilarity relation over a finite Kripke structure can be computed in polynomial time (with respect to the size of the structure).

The key observations

Algorithm 1

Lemma

The bisimilarity relation over a finite Kripke structure can be computed in polynomial time (with respect to the size of the structure).

Let n be the size of the Kripke structure.

The key observations

1. What is the bound on i (the moment when the approximants converge).

Algorithm 1

Lemma

The bisimilarity relation over a finite Kripke structure can be computed in polynomial time (with respect to the size of the structure).

Let n be the size of the Kripke structure.

The key observations

1. In B_0 there is n^2 pairs and if $\forall_{j<i} B_j \supset B_{j+1}$ then $i \leq n^2$.

Algorithm 1

Lemma

The bisimilarity relation over a finite Kripke structure can be computed in polynomial time (with respect to the size of the structure).

Let n be the size of the Kripke structure.

The key observations

1. In B_0 there is n^2 pairs and if $\forall_{j<i} B_j \supset B_{j+1}$ then $i \leq n^2$.
2. What is the complexity of calculating a single approximant?

Algorithm 1

Lemma

The bisimilarity relation over a finite Kripke structure can be computed in polynomial time (with respect to the size of the structure).

Let n be the size of the Kripke structure.

The key observations

1. In B_0 there is n^2 pairs and if $\forall_{j<i} B_j \supset B_{j+1}$ then $i \leq n^2$.
2. A single approximant in time (very naive) $n^2 \cdot |E|^2$.

THE PARTITION REFINEMENT ALGORITHM

ROBERT PAIGE AND ROBERT E. TARJAN

A coarsest partition.

- A given directed graph (V, E).
- $P \subset \mathbb{P}(V)$ is a partition of V iff
  1. $\forall_{S,T \in P} S \cap T = \emptyset$,
  2. $\bigcup_{S \in P} S = V$.

A coarsest partition.

- A given directed graph (V, E).
- $P \subset \mathbb{P}(V)$ is a partition of V iff
  1. $\forall_{S,T \in P} S \cap T = \emptyset$,
  2. $\bigcup_{S \in P} S = V$.
- A partition P is a refinement of a partition R if $\forall_{S \in P} \exists_{T \in R} S \subseteq T$.

A coarsest partition.

- A given directed graph (V, E).
- $P \subset \mathbb{P}(V)$ is a partition of V iff
  1. $\forall_{S,T \in P} S \cap T = \emptyset$,
  2. $\bigcup_{S \in P} S = V$.
- A partition P is a refinement of a partition R if $\forall_{S \in P} \exists_{T \in R} S \subseteq T$.
- Let $S \subseteq V$, $E^{-1}(S) \stackrel{\text{def}}{=} \{x \in V : \exists_{y \in S} x \to y\}$.

A coarsest partition.

- A given directed graph (V, E).
- $P \subset \mathbb{P}(V)$ is a partition of V iff
  1. $\forall_{S,T \in P} S \cap T = \emptyset$,
  2. $\bigcup_{S \in P} S = V$.
- A partition P is a refinement of a partition R if $\forall_{S \in P} \exists_{T \in R} S \subseteq T$.
- Let $S \subseteq V$, $E^{-1}(S) \stackrel{\text{def}}{=} \{x \in V : \exists_{y \in S} x \to y\}$.
- We say that the set S is stable with respect to the set T if $S \subseteq E^{-1}(T) \vee \emptyset = S \cap E^{-1}(T)$.
- We say that the partition P is stable with respect to the set T if $\forall_{S \in P} S$ is stable with respect to T.

A coarsest partition.

- A given directed graph (V, E).
- $P \subset \mathbb{P}(V)$ is a partition of V iff
  1. $\forall_{S,T \in P} S \cap T = \emptyset$,
  2. $\bigcup_{S \in P} S = V$.
- A partition P is a refinement of a partition R if $\forall_{S \in P} \exists_{T \in R} S \subseteq T$.
- Let $S \subseteq V$, $E^{-1}(S) \stackrel{\text{def}}{=} \{x \in V : \exists_{y \in S} x \to y\}$.
- We say that the set S is stable with respect to the set T if $S \subseteq E^{-1}(T) \vee \emptyset = S \cap E^{-1}(T)$.
- We say that the partition P is stable with respect to the set T if $\forall_{S \in P} S$ is stable with respect to T.
- We say that the partition P is stable if it is stable with respect to every set in P.

Coarsest stable partition problem

Input:	A graph $G = (V, E)$ and its partition P.
Question:	Find a coarsest stable partition R that is a refinement of P.

Coarsest stable partition problem

Input:	A graph $G = (V, E)$ and its partition P.
Question:	Find a coarsest stable partition R that is a refinement of P.

Lemma

Let K be a Kripke structure and P a partition of its vertices according to labelling with predicates L i.e. $\forall_{S \in P} \forall_{s \in S} \forall_{t \in V} t \in S \iff L(s) = L(t)$. Then the coarsest stable partition R refining P defines the bisimilarity relation for K.

Coarsest stable partition problem

Input:	A graph $G = (V, E)$ and its partition P.
Question:	Find a coarsest stable partition R that is a refinement of P.

Lemma

Let K be a Kripke structure and P a partition of its vertices according to labelling with predicates L i.e. $\forall_{S \in P} \forall_{s \in S} \forall_{t \in V} t \in S \iff L(s) = L(t)$. Then the coarsest stable partition R refining P defines the bisimilarity relation for K.

Proof.

1. Any stable refinement of P defines an equivalence relation which is a bisimulation.

$\square$

Coarsest stable partition problem

Input:	A graph $G = (V, E)$ and its partition P.
Question:	Find a coarsest stable partition R that is a refinement of P.

Lemma

Let K be a Kripke structure and P a partition of its vertices according to labelling with predicates L i.e. $\forall_{S \in P} \forall_{s \in S} \forall_{t \in V} t \in S \iff L(s) = L(t)$. Then the coarsest stable partition R refining P defines the bisimilarity relation for K.

Proof.

1. Any stable refinement of P defines an equivalence relation which is a bisimulation.
2. Any equivalence relation which is a bisimulation defines a stable refinement of P.

$\square$

Coarsest stable partition problem

Input:	A graph $G = (V, E)$ and its partition P.
Question:	Find a coarsest stable partition R that is a refinement of P.

Lemma

Let K be a Kripke structure and P a partition of its vertices according to labelling with predicates L i.e. $\forall_{S \in P} \forall_{s \in S} \forall_{t \in V} t \in S \iff L(s) = L(t)$. Then the coarsest stable partition R refining P defines the bisimilarity relation for K.

Proof.

1. Any stable refinement of P defines an equivalence relation which is a bisimulation.
2. Any equivalence relation which is a bisimulation defines a stable refinement of P.
3. The bisimilarity defines a coarsest partition.

$\square$

Algorithm 2

Definition - Split operation

- Let $split(S, Q)$ be the refinement of Q obtained by replacing each block $B \in Q$ such that $B \cap E^{-1}(S) \neq \emptyset \wedge B - E^{-1}(S) \neq \emptyset$ by the two blocks $B' = B \cap E^{-1}(S)$ and $B'' = B - E^{-1}(S)$.

Algorithm 2

Definition - Split operation

- Let $split(S, Q)$ be the refinement of Q obtained by replacing each block $B \in Q$ such that $B \cap E^{-1}(S) \neq \emptyset \wedge B - E^{-1}(S) \neq \emptyset$ by the two blocks $B' = B \cap E^{-1}(S)$ and $B'' = B - E^{-1}(S)$.
- We call S a splitter of Q if $split(S, Q) \neq Q$.

Algorithm 2

Definition - Split operation

- Let $split(S, Q)$ be the refinement of Q obtained by replacing each block $B \in Q$ such that $B \cap E^{-1}(S) \neq \emptyset \wedge B - E^{-1}(S) \neq \emptyset$ by the two blocks $B' = B \cap E^{-1}(S)$ and $B'' = B - E^{-1}(S)$.
- We call S a splitter of Q if $split(S, Q) \neq Q$.
- Q is unstable with respect to S if and only if S is a splitter of Q.

Algorithm 2

Definition - Split operation

- Let $split(S, Q)$ be the refinement of Q obtained by replacing each block $B \in Q$ such that $B \cap E^{-1}(S) \neq \emptyset \wedge B - E^{-1}(S) \neq \emptyset$ by the two blocks $B' = B \cap E^{-1}(S)$ and $B'' = B - E^{-1}(S)$.
- We call S a splitter of Q if $split(S, Q) \neq Q$.
- Q is unstable with respect to S if and only if S is a splitter of Q.

New algorithm, first try.

1. Let Q_0 be a partition of the states of the Kripke structure along the labels.
2. Let $S_i \in Q_i$ be a set such that Q_i is unstable with respect to S_i. If it does not exist then return Q_i.
3. $Q_{i+1} = split(S_i, Q_i)$. Go to point 2.

Algorithm 2

New algorithm, first try. Is it correct?

1. Let Q_0 be a partition of the states of the Kripke structure along the labels.
2. Let $S_i \in Q_i$ be an set such that Q_i is unstable with respect to S_i. If it does not exist then return Q_i.
3. $Q_{i+1} = split(S_i, Q_i)$. Go to point 2.

New algorithm, first try. Is it correct?

1. It returns a refinement of the initial partition Q_0.
2. It is stable with respect to each its element.
3. The procedure guaranties that it is coarsest partition.
   The proof is by showing that the coarsest partitions before and after every split operations are the same.

Algorithm 2, determinization and analysis.

New algorithm, first try. The complexity?

1. Let Q_0 be a partition of the states of the Kripke structure along the labels.
2. Find $S_i \in Q_i$ such that Q_i is unstable with respect to S_i. If it does not exist then return Q_i.
3. $Q_{i+1} = split(S_i, Q_i)$. Go to point 2.

Analysis.

Algorithm 2, determinization and analysis.

New algorithm, first try. The complexity?

1. Let Q_0 be a partition of the states of the Kripke structure along the labels.
2. Find $S_i \in Q_i$ such that Q_i is unstable with respect to S_i. If it does not exist then return Q_i.
3. $Q_{i+1} = split(S_i, Q_i)$. Go to point 2.

Analysis.

1. What is the bound on i?

Algorithm 2, determinization and analysis.

New algorithm, first try. The complexity?

1. Let Q_0 be a partition of the states of the Kripke structure along the labels.
2. Find $S_i \in Q_i$ such that Q_i is unstable with respect to S_i. If it does not exist then return Q_i.
3. $Q_{i+1} = split(S_i, Q_i)$. Go to point 2.

Analysis.

1. What is the bound on i? $|V|$.
2. Can we look for S_i in time proportional to $|E|$?
3. Can we split in time proportional to $|E|$?.

Algorithm 2, determinization and analysis.

New algorithm, first try. The complexity?

1. Let Q_0 be a partition of the states of the Kripke structure along the labels.
2. Find $S_i \in Q_i$ such that Q_i is unstable with respect to S_i. If it does not exist then return Q_i.
3. $Q_{i+1} = split(S_i, Q_i)$. Go to point 2.

Analysis.

1. What is the bound on i? $|V|$.
2. Can we look for S_i in time proportional to $|E|$?
3. Can we split in time proportional to $|E|$?.
4. It works in $|E| \cdot |V|$.

Algorithm 2, data structures.

All lists are two way linked lists.

We have states, blocks, a list of blocks in the partition Q_i.

Every state remembers

- pointers to representations of that vertex on every list that contain it,
- a list of incoming edges,
- its block.

Every block remembers

- pointers to representations of that block on every list that contain it,
- the list of its states,
- the number of states.

The Page and Tarjan algorithm.

The complexity $|E| \cdot log(|V|)$.

Main ideas:

The Page and Tarjan algorithm.

The complexity $|E| \cdot log(|V|)$.

Main ideas:

1. The algorithm is working in time proportional to the number of operations on edges.

The Page and Tarjan algorithm.

The complexity $|E| \cdot log(|V|)$.

Main ideas:

1. The algorithm is working in time proportional to the number of operations on edges.
2. Instead of splitters the algorithm uses candidates for splitters. A candidate sometimes does not split anything but can be found in a constant time.

The Page and Tarjan algorithm.

The complexity $|E| \cdot log(|V|)$.

Main ideas:

1. The algorithm is working in time proportional to the number of operations on edges.
2. Instead of splitters the algorithm uses candidates for splitters. A candidate sometimes does not split anything but can be found in a constant time.
3. Every vertex belongs to a candidate for a splitter at most $log(|V|)$ number of times.

The Page and Tarjan algorithm.

The complexity $|E| \cdot log(|V|)$.

Main ideas:

1. The algorithm is working in time proportional to the number of operations on edges.
2. Instead of splitters the algorithm uses candidates for splitters. A candidate sometimes does not split anything but can be found in a constant time.
3. Every vertex belongs to a candidate for a splitter at most $log(|V|)$ number of times.
4. Every edge is processed only if it ends in the current candidate for a splitter.

The Page and Tarjan algorithm.

The complexity $|E| \cdot log(|V|)$.

Main ideas:

1. The algorithm is working in time proportional to the number of operations on edges.
2. Instead of splitters the algorithm uses candidates for splitters. A candidate sometimes does not split anything but can be found in a constant time.
3. Every vertex belongs to a candidate for a splitter at most $log(|V|)$ number of times.
4. Every edge is processed only if it ends in the current candidate for a splitter.

Why this gives us $|E| \cdot log(|V|)$?

New refine strategy (two splits at once).

New refine strategy (two splits at once).

1. Let X and Y are partitions such that:
   - Y is a refinement of X,
   - Y is stable with respect to X.

New refine strategy (two splits at once).

1. Let X and Y are partitions such that:
   - Y is a refinement of X,
   - Y is stable with respect to X.
2. Find a block $S \in X$ that is not a block of Y.

New refine strategy (two splits at once).

1. Let X and Y are partitions such that:
   - Y is a refinement of X,
   - Y is stable with respect to X.
2. Find a block $S \in X$ that is not a block of Y.
3. Find a block $B \in Y$ such that $B \in S$ and $|B| \leq \frac{|S|}{2}$.

New refine strategy (two splits at once).

1. Let X and Y are partitions such that:
   - Y is a refinement of X,
   - Y is stable with respect to X.
2. Find a block $S \in X$ that is not a block of Y.
3. Find a block $B \in Y$ such that $B \in S$ and $|B| \leq \frac{|S|}{2}$.
4.
   - $X := X - S + B + (S \setminus B)$
   - $Y := split(S - B, split(B, Y))$.

New refine strategy (two splits at once).

1. Let X and Y are partitions such that:
   - Y is a refinement of X,
   - Y is stable with respect to X.
2. Find a block $S \in X$ that is not a block of Y.
3. Find a block $B \in Y$ such that $B \in S$ and $|B| \leq \frac{|S|}{2}$.
4.
   - $X := X - S + B + (S \setminus B)$
   - $Y := split(S - B, split(B, Y))$.
5. Observe that:
   - Y is a refinement of X,
   - Y is stable with respect to X.

New refine strategy (two splits at once).

1. Let X and Y are partitions such that:
   - Y is a refinement of X,
   - Y is stable with respect to X.
2. Find a block $S \in X$ that is not a block of Y.
3. Find a block $B \in Y$ such that $B \in S$ and $|B| \leq \frac{|S|}{2}$.
4.
   - $X := X - S + B + (S \setminus B)$
   - $Y := split(S - B, split(B, Y))$.
5. Observe that:
   - Y is a refinement of X,
   - Y is stable with respect to X.

The algorithm (high level)

Take $X = V$ and Y being an initial partition. Use new refine strategy until $X \neq Y$.

New refine strategy (two splits at once).

1. Let X and Y are partitions such that:
   - Y is a refinement of X,
   - Y is stable with respect to X.
2. Find a block $S \in X$ that is not a block of Y.
3. Find a block $B \in Y$ such that $B \in S$ and $|B| \leq \frac{|S|}{2}$.
4.
   - $X := X - S + B + (S \setminus B)$
   - $Y := split(S - B, split(B, Y))$.
5. Observe that:
   - Y is a refinement of X,
   - Y is stable with respect to X.

The algorithm (high level)

Take $X = V$ and Y being an initial partition. Use new refine strategy until $X \neq Y$.

- How many times each vertex can be an element of B a candidate for splitter?

2 Find a block $S \in X$ that is not a block of Y.

3 Find a block $B \in Y$ such that $B \subset S$ and $|B| \leq \frac{|S|}{2}$.

4
- $X := X - S + B + (S \setminus B)$
- $Y := split(S - B, split(B, Y))$.

2 Find a block $S \in X$ that is not a block of Y.

Keep a list of compound blocks of X.

3 Find a block $B \in Y$ such that $B \subset S$ and $|B| \leq \frac{|S|}{2}$.

4
- $X := X - S + B + (S \setminus B)$
- $Y := split(S - B, split(B, Y))$.

2 Find a block $S \in X$ that is not a block of Y.

Keep a list of compound blocks of X.

3 Find a block $B \in Y$ such that $B \subset S$ and $|B| \leq \frac{|S|}{2}$.

For each $S \in X$ we store a list of blocks to which it is refined in Y.
For each block of $B \in Y$ we store a number of its elements.

4
- $X := X - S + B + (S \setminus B)$
- $Y := split(S - B, split(B, Y))$.

2 Find a block $S \in X$ that is not a block of Y.

Keep a list of compound blocks of X.

3 Find a block $B \in Y$ such that $B \subset S$ and $|B| \leq \frac{|S|}{2}$.

For each $S \in X$ we store a list of blocks to which it is refined in Y.
For each block of $B \in Y$ we store a number of its elements.

4
- $X := X - S + B + (S \setminus B)$
- $Y := split(S - B, split(B, Y))$.

Lemma

$split(S - B, split(B, Y))$ can be computed in a time proportional to number of edges incoming to B.

2 Find a block $S \in X$ that is not a block of Y.

Keep a list of compound blocks of X.

3 Find a block $B \in Y$ such that $B \subset S$ and $|B| \leq \frac{|S|}{2}$.

For each $S \in X$ we store a list of blocks to which it is refined in Y.
For each block of $B \in Y$ we store a number of its elements.

4
- $X := X - S + B + (S \setminus B)$
- $Y := split(S - B, split(B, Y))$.

Lemma

$split(S - B, split(B, Y))$ can be computed in a time proportional to number of edges incoming to B.

To understand what are the precise data structures (lists and records) stored to maintain all needed information please go to pages 9 (the last paragraph), 10 and 11 of the Tarjan paper (in bibliography).

Bibliography

1. The algorithm:
   Robert Paige and Robert E. Tarjan. 1987. Three partition refinement algorithms. SIAM J. Comput. 16, 6 (December 1987), 973-989. DOI=10.1137/0216062 http://dx.doi.org/10.1137/0216062
   `ftp://ftp.cs.princeton.edu/reports/1986/038.pdf`
2. The algorithm for branching bisimulation:
   `http://www.win.tue.nl/~jfg/articles/mlogn_branching_algorithm.pdf`