Algorithms for Strong Bisimilarity on Finite Transition Systems

COMP6752 Seminar

Joshua Lau
University of New South Wales, Sydney

May 7, 2018
Outline

1. Definitions

2. The fixed-point approach

3. An $O(nm)$ algorithm for strong bisimilarity

4. An $O(m \log n)$ algorithm for strong bisimilarity on single action LTS’

5. Handling multiple actions
We call an LTS $\mathcal{L} = (P, \Sigma, \rightarrow)$ finite if it has finitely many states and transitions.

- Let $n = |P|$ denote the number of states.
- Let $\Sigma = \{a_1, a_2, \ldots, a_{|\Sigma|}\}$ be the set of actions. Note $|\Sigma| \leq m$.
- Let $m = |\rightarrow|$ denote the number of transitions, where $\rightarrow \subseteq P \times \Sigma \times P$.

Note that there is no special $\tau$ action, as we will only deal with the notion of strong bisimulation (referred to simply as bisimulation).

We assume $n \leq m + 1$ and thus $n = O(m)$. If not, we can fuse all states with no outgoing transitions.
Partitions

Let $\pi \subseteq 2^P$.

- $\pi$ is a *partition* of $P$ if and only if $P = \bigcup \pi$.
- Each element of $\pi$ is called a *block*.
- $\pi$ induces an equivalence relation on $P$.
- Given two partitions / equivalence relations $a, b$ we say $a \subseteq b$ (or $a$ *refines* $b$) if the equivalence relation induced by $a$ is contained in the equivalence relation induced by $b$.

Suppose $P = \{1, 2, 3, 4, 5\}$.

- $\{\{1, 3\}, \{4\}, \{2, 5\}\}$ is a partition of $P$.
- $\{\{1, 3\}, \{3, 4\}, \{2, 5\}\}$ is not a partition of $P$.
- $\{\{1\}, \{3\}, \{4\}, \{2, 5\}\} \subseteq \{\{1, 3\}, \{2, 4, 5\}\}$.

Refinement can be thought of as splitting blocks.
A relation \( R \subseteq P \times P \) is a **bisimulation** if every \((s, t) \in R\) satisfies the following properties:

- If \( s \xrightarrow{a} s' \) then there is some \( t' \) such that \( t \xrightarrow{a} t' \) and \((s', t') \in R\).
- If \( t \xrightarrow{a} t' \) then there is some \( s' \) such that \( s \xrightarrow{a} s' \) and \((s', t') \in R\).

Note that the empty relation is a bisimulation.

We say that two states \( s \) and \( t \) are **bisimilar** or **bisimulation equivalent**, written \( s \sim t \), if and only if there is some bisimulation containing \((s, t)\).
Given two process graphs, we would like to decide if they are bisimilar.

1. Combine the states / transitions of each of the process graphs to form one LTS.

2. The process graphs are bisimilar if and only if the initial states are related by $\sim$.

Fusing bisimilar states also gives the smallest process graph bisimilar to the original (the minimisation problem).
Outline

1. Definitions

2. The fixed-point approach

3. An $O(nm)$ algorithm for strong bisimilarity

4. An $O(m \log n)$ algorithm for strong bisimilarity on single action LTS’s

5. Handling multiple actions
Intuition
Intuition
Intuition
Intuition
Intuition
The fixed-point approach

Idea:

1. Start by approximating $\sim$ by identifying all states.
2. Refine our approximation by looking “one move ahead”: that is, identify states which can reach equivalent states in a single move for every action.
3. Repeat until we can’t refine any further.

Let $R \subseteq P \times P$ be a relation on the states of $P$. Then define

$$\mathcal{F}_\mathcal{L}(R) = \{(s, t) | (s \xrightarrow{a} s' \implies \exists t'. t \xrightarrow{a} t' \land (s', t') \in R) \text{ and } (t \xrightarrow{a} t' \implies \exists s'. s \xrightarrow{a} s' \land (s', t') \in R)\}$$

Remark

Note that a relation is a bisimulation if and only if it is a fixed point of $\mathcal{F}_\mathcal{L}$.
The fixed-point approach

Theorem

$F_L$ has a greatest fixed point, which is $\sim$.

Proof.

$F_L$ is a monotonic function over the complete lattice $(2^{P \times P}, \subseteq)$, so we can apply the Knaster-Tarski Theorem. Hence, a greatest fixed point exists, and is a bisimulation.

Every bisimulation is a fixed point of $F_L$ and is included in the greatest fixed point. Hence, the greatest fixed point is $\sim$. 

$\square$
The fixed-point approach

Definition

Start with the relation $R_0 = P \times P$, and define $R_i = \mathcal{F}_\mathcal{L}^i(R_0) = \mathcal{F}_\mathcal{L}(R_{i-1})$ for all $i > 0$. Then we obtain that $(s, t) \in R_i$ if and only if, for every $a \in \Sigma$:

- If $s \xrightarrow{a} s'$, then there is $t'$ such that $t \xrightarrow{a} t'$ and $(s', t') \in R_{i-1}$
- If $t \xrightarrow{a} t'$, then there is $s'$ such that $s \xrightarrow{a} s'$ and $(s', t') \in R_{i-1}$

From the definition, it can be shown that:

- Each $R_i$ is an equivalence relation.
- $R_i \subseteq R_{i-1}$ for all $i > 0$. 
The fixed-point approach

**Theorem (Hennessy and Milner, 1985)**

If $\mathcal{L}$ is (image-)finite, then $s \sim t$ if and only if $(s, t) \in R_i$ for all $i \in \mathbb{N}$.

**Proof.**

Omitted, but stems from the fact that if $\mathcal{L}$ is (image-)finite, then $\mathcal{F}_\mathcal{L}$ is anticontinuous.

**Algorithm**

Compute $R_0, R_1, \ldots$ until $R_{i-1} = R_i$. Then $R_i \equiv \sim$ is the greatest fixed point we desire.
The fixed-point approach

Algorithm

Compute $R_0, R_1, \ldots$ until $R_{i-1} = R_i$. Then $R_i \equiv \sim$ is the greatest fixed point we desire.

- Each application of $\mathcal{F}_\mathcal{L}$ to $R_j$ ($j < i - 1$) splits one or more blocks.
- This can happen at most $n - 1$ times.
- Naively, we can compute $\mathcal{F}_\mathcal{L}(R_j)$ in $O(m^2)$ time.

This gives an $O(nm^2)$ partition refinement algorithm for bisimulation!

Remark

Whenever $R_0$ is an equivalence relation, the algorithm computes the coarsest refinement of $R_0$ that is also a bisimulation.

Can we implement this partition refinement idea faster?
Outline

1. Definitions

2. The fixed-point approach

3. An $O(nm)$ algorithm for strong bisimilarity

4. An $O(m \log n)$ algorithm for strong bisimilarity on single action LTS

5. Handling multiple actions
An $O(nm)$ algorithm for strong bisimilarity

Due to Kanellakis and Smolka (1983).

Let $B$ be a block in a partition $\pi$, and $a \in \Sigma$ be some action. We can assign each block in $\pi$ a numerical ID from 1 to $|\pi|$. Then define $actionSplit(B, a, \pi)$ as follows:

**Definition ($actionSplit(B, a, \pi)$)**

1. If $|B| = 1$, then $B$ can’t be split. Otherwise, let $s$ be any state in $B$.
2. Find the set of blocks (by ID) that $s$ can reach directly (in one step) through some $a$ transition.
3. Form two sub-blocks out of the states in $B$: those that can reach precisely the same set as $s$, and those that can’t. These sub-blocks partition (split) $B$.
4. If one of the blocks is empty, $B$ can’t be split by $a$. Otherwise, return the two sub-blocks.
Example
Example
Example
Example
Remark

Intuitively, an $\text{actionSplit}(B, a, \pi)$ performs part of the “single step” of $\mathcal{F}_L$.

Suppose the $a$ transitions from each state are sorted by the block ID of the destination, for each $a \in \Sigma$.

Then, we can perform $\text{actionSplit}(B, a, \pi)$ in $O(m(B, a))$ time, where $m(B, a)$ is the total number of outgoing $a$ transitions from states in $B$.

1. De-duplicate ($\text{uniq}$) each list of states.
2. Two sorted lists are equal if and only if their lengths are equal and corresponding elements are equal.
An $O(nm)$ algorithm for strong bisimilarity

Algorithm

Start with $\pi_0 = \{P\}$. At each step $i$, find a block $B \in \pi_i$, and an action $a \in \Sigma$ so that we can $\text{actionSplit}(B, a, \pi_i)$. If none, then $\sim = \pi_i$ and we are done. Otherwise, replace $B$ in $\pi_i$ with the returned sub-blocks, forming $\pi_{i+1}$. Repeat.

- At the beginning of each step, we need to (re-)sort the transitions, since states now belong to different blocks.
- We can do this in $O(n + m + |\Sigma|) = O(m)$ using Radix sort.
- We iterate over all blocks and actions, but each transition is processed in precisely one block. Thus, finding and performing an $\text{actionSplit}$ takes $O(m)$ time.
- We can split a partition at most $n - 1$ times, so the overall time complexity is $O(nm)$. 
An $O(nm)$ algorithm for strong bisimilarity

Proof.

From earlier, we know that $\sim \subseteq R_i$ for all $i$. By induction, we can show that $R_i \subseteq \pi_i$: it is true initially, and each actionSplit performs only part of a single step, so it splits as most as much as a single step. Hence, $\sim \subseteq \pi_i$ for all $i$.

Suppose we are unable to actionSplit $\pi_j$. It follows that $\pi_j$ is a fixed point of $\mathcal{F}_L$: if we cannot perform part of a single step, then we can not perform a single step at all. Hence, $\pi_j \subseteq \sim$, since $\sim$ is the greatest fixed point, so $\sim = \pi_j$. □
Outline

1. Definitions
2. The fixed-point approach
3. An $O(nm)$ algorithm for strong bisimilarity
4. An $O(m \log n)$ algorithm for strong bisimilarity on single action LTS
5. Handling multiple actions
Paige-Tarjan Algorithm

Due to *Paige and Tarjan* (1986).

Restrict ourselves to LTS’ with a single action.
Stability

Let $B$ be a block in some partition $\pi$, and let $S \subseteq P$ be some subset of states.

- We say that $B$ is stable w.r.t. $S$ if and only if every $s \in B$ has a transition to some $s' \in S$, or none of them do.
- $\pi$ is stable w.r.t. $S$ if and only if every $B \in \pi$ is.
- $\pi$ is stable w.r.t. another partition $\pi'$ if and only if $\pi$ is stable w.r.t. each $B' \in \pi'$.

Remark

The equivalence relation induced by a partition $\pi$ is a bisimulation if and only if $\pi$ is stable w.r.t. itself.
Example

Green stable w.r.t. orange.
Green **stable** w.r.t. orange.
Orange **not stable** w.r.t. green.
Green stable w.r.t. orange.
Orange stable w.r.t. green.
Stability

Theorem

1. **Stability is preserved by refinement:** if $\pi \subseteq \pi'$ and $\pi'$ is stable w.r.t. $S$ then so is $\pi$.

2. **Stability is preserved by union:** if $\pi$ is stable w.r.t. $S$, and also w.r.t. $T$, then $\pi$ is stable w.r.t. $S \cup T$.

Proof.

1. The condition for stability is only weakened by refinement, since the blocks in $\pi$ are subsets of the blocks in $\pi'$.

2. Let $B$ be a block in $\pi$. If every state in $B$ has a transition to some state in $S$, then every state also has a transition to $S \cup T$. Otherwise, w.l.o.g., no state has a transition to $S$ or $T$, as required.
Splitting

Definition

Define $\text{split}(\pi, S)$ as follows: for each block $B \in \pi$, we separate $B$ into two sub-blocks: those that have transitions to $S$, and those which do not. Then, return the partition formed by all the sub-blocks, removing any empty ones.

Theorem

1. $\text{split}(\pi, S) \subseteq \pi.$
2. If $\pi$ is stable w.r.t. $S$, then $\text{split}(\pi, S) = \pi.$
3. $\text{split}(\pi, S)$ is always stable w.r.t. $S.$

Proof.

Straightforward from the definition.
Example

\[ \text{split}(\pi, \text{Green}) \]

\[ \pi = \{ \text{Green, Orange} \}. \]
Example

\[ \text{split}(\pi, \text{Green}) \]

\[ \pi = \{\text{Green, Orange}\}. \]

Green stable w.r.t. Green, but Orange not stable w.r.t. Green.
Example

\( \text{split}(\pi, \text{Green}) \)

\[
\pi = \{ \text{Green}, \text{Orange}, \text{Blue} \}.
\]
Example

\[ \text{split}(\pi, \text{Green}) \]

\[ \pi = \{ \text{Green, Orange, Blue} \}. \]

All blocks stable w.r.t. each other, so \( \pi \) stable w.r.t. \( \pi \).
We can formulate a (slow) algorithm for bisimilarity as follows: start with $\pi = \{P\}$. Then, while $\pi$ is not stable w.r.t. itself, choose some block $B \in \pi$ so that $\pi$ is not stable w.r.t. $B$, and replace $\pi$ with $\text{split}(\pi, B)$.

Proof.
Similar to that of the Kanellakis-Smolka algorithm: each $\text{split}$ performs part of a “single step” (this time, based on destination block rather than source and action).
Algorithm

We can formulate a somewhat equivalent algorithm as follows: start with \( \pi = \{P\} \). Then, while \( \pi \) is not stable w.r.t. itself, choose some union of blocks \( S \) in \( \pi \) so that \( \pi \) is not stable w.r.t. \( S \), and replace \( \pi \) with \( \text{split}(\pi, S) \).

Sketch Proof.

Since we can still split based on single blocks, it remains to show that splitting based on a union of blocks doesn’t cause \( \pi \) to be finer than necessary. We observe that \( \text{split}(\text{split}(\pi, A), B) \subseteq \text{split}(\pi, A \cup B) \), and the result follows.
Three-way Splitting

Suppose $\pi$ is stable w.r.t. some set $S$, and let $B \subseteq S$.

**Key idea:** We can efficiently compute $\pi'' = \text{split}(\text{split}(\pi, B), S \setminus B)$.

Let $C$ be a block in $\pi$.

- If $C$ has no transitions to $S$, $C$ appears unchanged in $\pi''$.
- Otherwise, every state in $C$ has a transition to $S$. In $\pi''$ these states are split into three sub-blocks:
  1. Those with transitions only to $B$.
  2. Those with transitions only to $S \setminus B$.
  3. Those with transitions to both $B$ and $S \setminus B$. 
Three-way Splitting

Let \( s \) be some state and \( \text{count}(s, B) \) denote the number of transitions from \( s \) to some state in \( B \). Then, since \( B \subseteq S \):

1. If \( \text{count}(s, B) = \text{count}(s, S) \) then \( s \) has transitions only to \( B \).
2. If \( \text{count}(s, B) = 0 \) then \( s \) has transitions only to \( S \setminus B \).
3. If \( 0 < \text{count}(s, B) < \text{count}(s, S) \) then \( s \) has transitions to both \( B \) and \( S \setminus B \).

**Takeaway**

- If we know \( \text{count}(s, B) \) and \( \text{count}(s, S) \) then we can decide which sub-block \( s \) belongs to in \( O(1) \).
- Hence, if we know \( \text{count}(s, S) \) for all \( s \), we can compute \( \pi'' \) in \( O(|B| + m^{-1}(B)) \), where \( m^{-1}(B) \) is the number of transitions into the states of \( B \).
We keep:

- A current partition $\pi$, representing our current approximation for $\sim$. Initially, $\pi$ contains two blocks: states with outgoing transitions, and those without.
- Another partition $X$, initially $\{P\}$. $X$ will dictate what splits we perform.
- $\text{count}(s, S)$ for all $s \in P$ and $S \in X$. We keep this sparsely, so the value is 0 if there is no entry.

We maintain the invariants:

1. $\pi \subseteq X$, that is, $\pi$ is always a refinement of $X$; and
2. $\pi$ is stable w.r.t. $X$. 
Paige-Tarjan Algorithm Invariants

1. \( \pi \subseteq X \), that is, \( \pi \) is always a refinement of \( X \); and
2. \( \pi \) is stable w.r.t. \( X \).

Call a block \( S \in X \) *simple*, if \( S \) is also in \( \pi \), and otherwise *compound*. It follows from the invariants that every compound block is the union of two or more blocks of \( \pi \).

Observe that when \( \pi = X \), \( \pi \) is stable w.r.t. itself and so \( \pi \) is a bisimulation. Otherwise, \( X \) contains at least one compound block.
Paige-Tarjan Algorithm

Repeat until $\pi = X$:

1. Pick any compound block $S \in X$.
2. Pick any block $B \in \pi$ such that $B \subseteq S$. Plainly, $B$ is one of the blocks of $\pi$ that make up $S$.
3. Replace $\pi$ with $\pi'' = \text{split}(\text{split}(\pi, B), S \setminus B)$, using three-way splitting in $O(|B| + m^{-1}(B))$.
4. Update $X$ with $X''$ formed by replacing $S$ with $B$ and $S \setminus B$.
5. Update count values, see next slide.

Return $\sim = \pi$. 
At the end of each iteration, we need to replace all \( \text{count}(s, S) \) values with \( \text{count}(s, B) \) and \( \text{count}(s, S \setminus B) \) values. To do this, we:

1. Recycle the \( \text{count}(s, S) \) values for \( \text{count}(s, S \setminus B) \) (e.g. by reusing the ID for \( S \) for \( S \setminus B \)).

2. Then, for each transition to a state in \( B \), decrement its \( \text{count}(s, S \setminus B) \) value, and increment its \( \text{count}(s, B) \) value.

Thus, we can update the counts in \( O(|B| + m^{-1}(B)) \).
We prove the algorithm by induction. Initially, since $X = \{P\}$, $\pi \subseteq X$. Also, $\pi$ is initially the blocks of states with transitions, and those without, which is precisely $\text{split}(P, P)$, so $\pi$ is initially stable w.r.t. $X$.

We need to show that if the invariants hold before each step, they also hold after each step:

1. Since $S$ was the union of blocks in $\pi$, and $B \in \pi$, we know that both $B$ and $S \setminus B$ were the union of blocks in $\pi$. Since $\pi''$ is $\pi$ after some splits, we see that $B$ and $S \setminus B$ must also be the union of (possibly smaller) blocks in $\pi''$. Hence, $\pi'' \subseteq X''$.

2. Initially, $\pi$ is stable w.r.t. $X$. $\pi''$ is stable w.r.t. $B$ and $S \setminus B$ since they were used for splitting. Every other block $B'$ in $X''$ was also in $X$, and so $\pi$ was stable w.r.t. $B'$. Hence, $\pi''$ is also stable w.r.t. $B'$ because $\pi'' \subseteq \pi$ so $\pi''$ is stable w.r.t. $X''$. 
Proof.

Since we split $X$ each time, we can make at most $n - 1$ splits, and thus the algorithm terminates.

Since we only ever split according to a union of blocks, this is merely an implementation of the previous algorithm, and correctness follows.
Paige-Tarjan Algorithm

How can we make sure this algorithm is fast?
Use plenty of pointers and doubly-linked lists!

Just kidding (kinda).

Remark

If we choose $B$ such that $|B| \leq |S|/2$ then:

- Each state appears in some “$B$-set” at most $\log_2 n + 1$ times.
- Each transition is considered only when its destination state is considered, which is at most $\log_2 n + 1$ times.

Hence, the algorithm is $O((n + m) \log n) = O(m \log n)$!

How do we choose such a $B$? Compare the sizes of the first two sets of $\pi$ that compose $S$: the smaller has the desired property.
Outline

1 Definitions

2 The fixed-point approach

3 An $O(nm)$ algorithm for strong bisimilarity

4 An $O(m \log n)$ algorithm for strong bisimilarity on single action LTS’

5 Handling multiple actions
Handling multiple actions

We reduce finding $\sim$ on multiple action LTS' to finding $\sim$ on a single action LTS.

\[ U \xrightarrow{a_i} V \]

\[ P_U \xrightarrow{} P_{U,a_i,V} \xrightarrow{} P_V \]
We use the same chain for all transitions, so we have
\( n + m + |\Sigma| + 1 = O(m) \) states, and \( 2m + |\Sigma| = O(m) \) transitions.

**Theorem**

\( U \sim V \) if and only if \( P_U \sim P_V \).

This gives an \( O(m \log m) \) algorithm for bisimilarity on LTS’s!

