Communication and Concurrency
Lecture 16

Colin Stirling (cps)

School of Informatics

14th November 2013
The (strong) bisimilarity problem

- **Given:** two processes $E$ and $F$
The (strong) bisimilarity problem

- **Given**: two processes $E$ and $F$
- **Decide**: is $E \sim F$? i.e., are $E$ and $F$ (strongly) bisimilar?
The (strong) bisimilarity problem

- Given: two processes $E$ and $F$
- Decide: is $E \sim F$? i.e., are $E$ and $F$ (strongly) bisimilar?
- Assume both $T_E$ and $T_F$ are finite
The (strong) bisimilarity problem

- **Given:** two processes $E$ and $F$
- **Decide:** is $E \sim F$? i.e., are $E$ and $F$ (strongly) bisimilar?
- **Assume both** $T_E$ and $T_F$ are finite
- **Observation:** whether $E \sim F$ depends only on $T_E$ and $T_F$
The (strong) bisimilarity problem

- **Given:** two processes $E$ and $F$
- **Decide:** is $E \sim F$? i.e., are $E$ and $F$ (strongly) bisimilar?
- **Assume both** $T_E$ and $T_F$ **are finite**
- **Observation:** whether $E \sim F$ depends only on $T_E$ and $T_F$
- **Restrict relations to subsets of** $S \times S$, where $S \subseteq S_E \cup S_F$. Notice that $S$ is finite
- **Outline of the algorithm:**

The (strong) bisimilarity problem

- **Given:** two processes $E$ and $F$
- **Decide:** is $E \sim F$? i.e., are $E$ and $F$ (strongly) bisimilar?
- **Assume both** $T_E$ and $T_F$ **are finite**
- **Observation:** whether $E \sim F$ depends only on $T_E$ and $T_F$
- **Restrict relations to subsets of** $S \times S$, where $S \subseteq S_E \cup S_F$. Notice that $S$ is finite
- **Outline of the algorithm:**
  - Compute $\sim \subseteq S \times S.$
The (strong) bisimilarity problem

- **Given:** two processes $E$ and $F$
- **Decide:** is $E \sim F$? i.e., are $E$ and $F$ (strongly) bisimilar?
- **Assume both** $T_E$ and $T_F$ are finite
- **Observation:** whether $E \sim F$ depends only on $T_E$ and $T_F$
- **Restrict relations to subsets of** $S \times S$, where $S \subseteq S_E \cup S_F$.
  Notice that $S$ is finite
- **Outline of the algorithm:**
  - Compute $\sim \subseteq S \times S$.
  - Check if $(E, F) \in \sim$. 
Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
Bisimilarity up to $n$

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
Bisimilarity up to $n$

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
- $E \sim_0 F$ for all $E$ and $F$. 

Bisimilarity up to $n$

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
  - $E \sim_0 F$ for all $E$ and $F$.
  - $E \sim_{n+1} F$ if and only if for every action $a$, $E \xrightarrow{a} E'$ implies $\exists F' : F \xrightarrow{a} F'$ such that $E' \sim_n F'$.
Bisimilarity up to $n$

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
  - $E \sim_0 F$ for all $E$ and $F$.
  - $E \sim_{n+1} F$ if and only if for every action $a$,
    - if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some $F'$ such that $E' \sim_n F'$, and
Bisimilarity up to $n$

- Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.
- For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:
- $E \sim_0 F$ for all $E$ and $F$.
- $E \sim_{n+1} F$ if and only if for every action $a$,
  - if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some $F'$ such that $E' \sim_n F'$, and
  - if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some $E'$ such that $E' \sim_n F'$.
Recall that $\sim$ is the largest bisimulation or the union of all bisimulations, and that it is a bisimulation itself.

For each $n \geq 0$, the relation $\sim_n$ between pairs of processes is inductively defined as follows:

- $E \sim_0 F$ for all $E$ and $F$.
- $E \sim_{n+1} F$ if and only if for every action $a$,
  - if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some $F'$ such that $E' \sim_n F'$, and
  - if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some $E'$ such that $E' \sim_n F'$.
Proposition For all $n \geq 0$,

1. $\sim_n \supseteq \sim$,
2. $\sim_n \supseteq \sim_{n+1}$, and
3. If $\sim_n = \sim_{n+1}$, then $\sim_n = \sim$. 

Proof: 1. By induction on $n$.

Base: $n = 0$. Trivial, because $E \sim_0 F$ for all $E$, $F$.

Step: Let $E \sim F$. We prove $E \sim_{n+1} F$.

Let $E \xrightarrow{a} E'$ be an arbitrary transition of $E$. Since $E \sim F$, there is a transition $F \xrightarrow{a} F'$ of $F$ such that $E' \sim F'$. By induction hypothesis, $E' \sim_n F'$.

Similarly we prove that for every transition $F \xrightarrow{a} F'$ of $F$ there is a transition $E \xrightarrow{a} E'$ of $E$ such that $E' \sim_n F'$.

By definition of $\sim_{n+1}$, we have $E \sim_{n+1} F$. 

Key result

**Proposition** For all $n \geq 0$,

1. $\sim_n \supseteq \sim$,
2. $\sim_n \supseteq \sim_{n+1}$, and
3. If $\sim_n = \sim_{n+1}$, then $\sim_n = \sim$.

➤ **Proof:** 1. By induction on $n$. 

Key result

Proposition For all $n \geq 0$,

1. $\sim_n \supseteq \sim$,
2. $\sim_n \supseteq \sim_{n+1}$, and
3. If $\sim_n = \sim_{n+1}$, then $\sim_n = \sim$.

Proof: 1. By induction on $n$.

Base: $n = 0$. Trivial, because $E \sim_0 F$ for all $E, F$.

Step: Let $E \sim F$. We prove $E \sim_{n+1} F$.
   Let $E \xrightarrow{a} E'$ be an arbitrary transition of $E$.
Key result

**Proposition** For all \( n \geq 0 \),

1. \( \sim_n \supseteq \sim \),
2. \( \sim_n \supseteq \sim_{n+1} \), and
3. If \( \sim_n = \sim_{n+1} \), then \( \sim_n = \sim \).

**Proof:**

1. By induction on \( n \).

**Base:** \( n = 0 \). Trivial, because \( E \sim_0 F \) for all \( E, F \).

**Step:** Let \( E \sim F \). We prove \( E \sim_{n+1} F \).

Let \( E \xrightarrow{a} E' \) be an arbitrary transition of \( E \).

Since \( E \sim F \), there is a transition \( F \xrightarrow{a} F' \) of \( F \) such that \( E' \sim F' \). By induction hypothesis, \( E' \sim_n F' \).
Key result

Proposition For all $n \geq 0$,

1. $\sim_n \supseteq \sim$,

2. $\sim_n \supseteq \sim_{n+1}$, and

3. If $\sim_n = \sim_{n+1}$, then $\sim_n = \sim$.

Proof: 1. By induction on $n$.

Base: $n = 0$. Trivial, because $E \sim_0 F$ for all $E, F$.

Step: Let $E \sim F$. We prove $E \sim_{n+1} F$.

Let $E \xrightarrow{a} E'$ be an arbitrary transition of $E$.

Since $E \sim F$, there is a transition $F \xrightarrow{a} F'$ of $F$ such that $E' \sim F'$. By induction hypothesis, $E' \sim_{n} F'$.

Similarly we prove that for every transition $F \xrightarrow{a} F'$ of $F$ there is a transition $E \xrightarrow{a} E'$ of $E$ such that $E' \sim_{n} F'$.

By definition of $\sim_{n+1}$, we have $E \sim_{n+1} F$. 
Proof of 2

2. $\sim_n \supseteq \sim_{n+1}$. By induction on $n$. 
Proof of 2

- 2. $\sim_n \supseteq \sim_{n+1}$. By induction on $n$.
- Base: $n = 0$. Trivial, because $E \sim_0 F$ for all $E, F$. 
Proof of 2

- 2. $\sim_n \supseteq \sim_{n+1}$. By induction on $n$.
- **Base:** $n = 0$. Trivial, because $E \sim_0 F$ for all $E, F$.
- **Step:** We assume $\sim_n \supseteq \sim_{n+1}$ and prove $\sim_{n+1} \supseteq \sim_{n+2}$.
Proof of 2

2. \( \sim_n \supseteq \sim_{n+1} \). By induction on \( n \).

Base: \( n = 0 \). Trivial, because \( E \sim_0 F \) for all \( E, F \).

Step: We assume \( \sim_n \supseteq \sim_{n+1} \) and prove \( \sim_{n+1} \supseteq \sim_{n+2} \).

Assume \( E \sim_{n+2} F \). We prove \( E \sim_{n+1} F \).

Let \( E \xrightarrow{a} E' \) be an arbitrary transition of \( E \).
Proof of 2

2. $\sim_n \supseteq \sim_{n+1}$. By induction on $n$.

Base: $n = 0$. Trivial, because $E \sim_0 F$ for all $E, F$.

Step: We assume $\sim_n \supseteq \sim_{n+1}$ and prove $\sim_{n+1} \supseteq \sim_{n+2}$

Assume $E \sim_{n+2} F$. We prove $E \sim_{n+1} F$.

Let $E \xrightarrow{a} E'$ be an arbitrary transition of $E$.

Since $E \sim_{n+2} F$, there is a transition $F \xrightarrow{a} F'$ of $F$ such that $E' \sim_{n+1} F'$. 

Proof of 2

2. $\sim_n \supseteq \sim_{n+1}$. By induction on $n$.

Base: $n = 0$. Trivial, because $E \sim_0 F$ for all $E, F$.

Step: We assume $\sim_n \supseteq \sim_{n+1}$ and prove $\sim_{n+1} \supseteq \sim_{n+2}$

Assume $E \sim_{n+2} F$. We prove $E \sim_{n+1} F$.

Let $E \xrightarrow{a} E'$ be an arbitrary transition of $E$.

Since $E \sim_{n+2} F$, there is a transition $F \xrightarrow{a} F'$ of $F$ such that $E' \sim_{n+1} F'$.

By induction hypothesis, $E' \sim_n F'$. 
2. $\sim_n \supseteq \sim_{n+1}$. By induction on $n$.

Base: $n = 0$. Trivial, because $E \sim_0 F$ for all $E, F$.

Step: We assume $\sim_n \supseteq \sim_{n+1}$ and prove $\sim_{n+1} \supseteq \sim_{n+2}$

Assume $E \sim_{n+2} F$. We prove $E \sim_{n+1} F$.
Let $E \xrightarrow{a} E'$ be an arbitrary transition of $E$.

Since $E \sim_{n+2} F$, there is a transition $F \xrightarrow{a} F'$ of $F$ such that $E' \sim_{n+1} F'$.

By induction hypothesis, $E' \sim_n F'$.

Similarly we prove that for every transition $F \xrightarrow{a} F'$ of $F$ there is a transition $E \xrightarrow{a} E'$ of $E$ such that $E' \sim_n F'$.
So $E \sim_{n+1} F$. 
3. If $\sim_n \equiv \sim_{n+1}$, then $\sim_n \equiv \sim$. 
Proof of 3

3. If \( \sim_n = \sim_{n+1} \), then \( \sim_n = \sim \).

We assume \( \sim_n = \sim_{n+1} \), and prove \( \sim_n = \sim \).
3. If $\sim_n = \sim_{n+1}$, then $\sim_n \equiv \sim$.

We assume $\sim_n = \sim_{n+1}$, and prove $\sim_n \equiv \sim$.

We have $\sim_n \supseteq \sim$ by (1).
Proof of 3

3. If $\sim_n = \sim_{n+1}$, then $\sim_n = \sim$.

We assume $\sim_n = \sim_{n+1}$, and prove $\sim_n = \sim$.

We have $\sim_n \supseteq \sim$ by (1).

To prove $\sim_n \subseteq \sim$, we show that $\sim_n$ is a bisimulation.
Proof of 3

3. If $\sim_n = \sim_{n+1}$, then $\sim_n = \sim$.

We assume $\sim_n = \sim_{n+1}$, and prove $\sim_n = \sim$.

We have $\sim_n \supseteq \sim$ by (1)

To prove $\sim_n \subseteq \sim$, we show that $\sim_n$ is a bisimulation.

Let $E \sim_n F$, and let $E \xrightarrow{a} E'$ be an arbitrary transition of $E$. Since $\sim_n = \sim_{n+1}$, we have $E \sim_{n+1} F$, and so there is a transition $F \xrightarrow{a} F'$ of $F$ such that $E' \sim_n F'$.
Proof of 3

3. If $\sim_n = \sim_{n+1}$, then $\sim_n = \sim$.

We assume $\sim_n = \sim_{n+1}$, and prove $\sim_n = \sim$.

We have $\sim_n \supseteq \sim$ by (1).

To prove $\sim_n \subseteq \sim$, we show that $\sim_n$ is a bisimulation.

Let $E \sim_n F$, and let $E \xrightarrow{a} E'$ be an arbitrary transition of $E$. Since $\sim_n = \sim_{n+1}$, we have $E \sim_{n+1} F$, and so there is a transition $F \xrightarrow{a} F'$ of $F$ such that $E' \sim_n F'$.

Similarly we prove that for every transition $F \xrightarrow{a} F'$ of $F$ there is a transition $E \xrightarrow{a} E'$ of $E$ such that $E' \sim_n F'$.

So $\sim_n$ is a bisimulation.
Scheme for the computation of \( \sim \)

- Compute \( \sim_0, \sim_1, \sim_2, \ldots \) until \( \sim_i = \sim_{i+1} \).
Scheme for the computation of $\sim$

- Compute $\sim_0, \sim_1, \sim_2, \ldots$ until $\sim_i = \sim_{i+1}$.
- Output $\sim_i$. 

Correctness: Part (3) of the Proposition.

Termination: Assume the procedure does not terminate. Then, by part (2) of the Proposition, we have an infinite chain $\sim_0 \supset \sim_1 \supset \sim_2 \ldots$ This contradicts the finiteness of $S$. 
Scheme for the computation of \( \sim \)

- Compute \( \sim_0, \sim_1, \sim_2, \ldots \) until \( \sim_i = \sim_{i+1} \).
- Output \( \sim_i \).
- **Correctness**: Part (3) of the Proposition.
Scheme for the computation of $\sim$

- Compute $\sim_0, \sim_1, \sim_2, \ldots$ until $\sim_i = \sim_{i+1}$.
- Output $\sim_i$.
- **Correctness:** Part (3) of the Proposition.
- **Termination:** Assume the procedure does not terminate. Then, by part (2) of the Proposition, we have an infinite chain

  $\sim_0 \supset \sim_1 \supset \sim_2 \ldots$

This contradicts the finiteness of $S$. 
Partition refinement algorithms

- **Idea:** think of $\sim$ not as a set of pairs, but as a set of equivalence classes.
Partition refinement algorithms

- **Idea:** think of $\sim$ not as a set of pairs, but as a set of equivalence classes.
- **Recall that** $\sim$ is an equivalence relation.

Proof sketch: Show that the elements of a partition satisfy this property if and only if they are the equivalence classes of a bisimulation. Show that the coarsest partition corresponds to $\sim$. 
Partition refinement algorithms

- **Idea:** think of $\sim$ not as a set of pairs, but as a set of equivalence classes.
- **Recall that** $\sim$ **is an equivalence relation**
- **Proposition:** $\sim$ **is the coarsest partition of** $S$ **satisfying the following property:** For every element $\{E_1, \ldots, E_k\} \subseteq S$ of the partition, and for every action $a$:

  - either none of $E_1, \ldots, E_k$ can do an $a$, or,
  - all of $E_1, \ldots, E_k$ can do an $a$, and there are processes $F_1, \ldots, F_k$ such that $E_i \xrightarrow{a} F_i$ for every $1 \leq i \leq k$, and moreover $\{F_1, \ldots, F_k\}$ is included in an element of the partition.

- **Proof sketch:** Show that the elements of a partition satisfy this property if and only if they are the equivalence classes of a bisimulation. Show that the coarsest partition corresponds to $\sim$. 
Partition refinement algorithms

- **Idea:** think of $\sim$ not as a set of pairs, but as a set of equivalence classes.

- **Recall that** $\sim$ **is an equivalence relation**

- **Proposition:** $\sim$ is the coarsest partition of $S$ satisfying the following property: For every element $\{E_1, \ldots E_k\} \subseteq S$ of the partition, and for every action $a$:
  - either none of $E_1, \ldots E_k$ can do an $a$, or,
Partition refinement algorithms

- **Idea:** think of $\sim$ not as a set of pairs, but as a set of equivalence classes.

- **Recall that** $\sim$ **is an equivalence relation**

- **Proposition:** $\sim$ **is the coarsest partition of** $S$ **satisfying the following property:** For every element $\{E_1, \ldots E_k\} \subseteq S$ of the partition, and for every action $a$:
  - either none of $E_1, \ldots E_k$ can do an $a$, or,
  - all of $E_1, \ldots E_k$ can do an $a$, and there are processes $F_1, \ldots, F_k$ such that $E_i \xrightarrow{a} F_i$ for every $1 \leq i \leq k$, and moreover $\{F_1, \ldots F_k\}$ is included in an element of the partition.

Proof sketch: Show that the elements of a partition satisfy this property if and only if they are the equivalence classes of a bisimulation. Show that the coarsest partition corresponds to $\sim$. 
Partition refinement algorithms

- **Idea:** think of \( \sim \) not as a set of pairs, but as a set of equivalence classes.

- **Recall that** \( \sim \) **is an equivalence relation**

- **Proposition:** \( \sim \) is the coarsest partition of \( S \) satisfying the following property: For every element \( \{E_1, \ldots, E_k\} \subseteq S \) of the partition, and for every action \( a \):
  - either none of \( E_1, \ldots, E_k \) can do an \( a \), or,
  - all of \( E_1, \ldots, E_k \) can do an \( a \), and there are processes \( F_1, \ldots, F_k \) such that \( E_i \xrightarrow{a} F_i \) for every \( 1 \leq i \leq k \), and moreover \( \{F_1, \ldots, F_k\} \) is included in an element of the partition.

- **Proof sketch:** Show that the elements of a partition satisfy this property if and only if they are the equivalence classes of a bisimulation.

Show that the coarsest partition corresponds to \( \sim \).
Splitting

Given two elements $P_1, P_2$ of a partition of $S$ and an action $a$, the result of splitting $P_1$ w.r.t $P_2$ and $a$ are the sets

\[ P_1' = \{ E \in P_1 \mid E \xrightarrow{a} F \text{ for some } F \in P_2 \} \]

\[ P_1'' = P_1 \setminus P_1' \]
Splitting

Given two elements $P_1, P_2$ of a partition of $S$ and an action $a$, the result of splitting $P_1$ w.r.t $P_2$ and $a$ are the sets

\[ P'_1 = \{ E \in P_1 \mid E \overset{a}{\rightarrow} F \text{ for some } F \in P_2 \} \]
\[ P''_1 = P_1 \setminus P'_1 \]

Input: $T_E, T_F$
Output: equivalence classes of $\sim$ on $S$

Initialize $\Pi := \{S\};$

Iterate: Choose an action $a$ and $P_1, P_2 \in \Pi$
Split $P_1$ with respect to $P_2$ and $a$;

\[ \Pi = (\Pi \setminus \{P_1\}) \cup \{P'_1, P''_1\}; \]
until a fixpoint is reached;

return $\Pi$
There are at most $|S| - 1$ splittings.
Complexity

- There are at most $|S| - 1$ splittings.
- Each splitting can be performed in time $O(|S| + |\delta|)$, where $\delta = \delta_E \cup \delta_F$ (complicated).
There are at most $|S| - 1$ splittings.

Each splitting can be performed in time $O(|S| + |\delta|)$, where $\delta = \delta_E \cup \delta_F$ (complicated).

So the running time is $O(|S| \cdot (|S| + |\delta|))$. 

Best known algorithm: $O(|\delta| \cdot \log(|S|))$. 

Complexity

- There are at most $|S| - 1$ splittings.
- Each splitting can be performed in time $O(|S| + |\delta|)$, where $\delta = \delta_E \cup \delta_F$ (complicated).
- So the running time is $O(|S| \cdot (|S| + |\delta|))$
- Best known algorithm: $O(|\delta| \cdot \log(|S|))$
The weak bisimilarity problem

- **Given:** two processes $E$ and $F$. 
The weak bisimilarity problem

- **Given**: two processes $E$ and $F$.
- **Decide**: is $E \approx F$? i.e., are $E$ and $F$ weakly bisimilar?
The weak bisimilarity problem

- **Given:** two processes $E$ and $F$.
- **Decide:** is $E \approx F$? i.e., are $E$ and $F$ weakly bisimilar?
- **Assume both** $T_E$ and $T_F$ are finite.
The weak bisimilarity problem

- **Given:** two processes $E$ and $F$.
- **Decide:** is $E \approx F$? i.e., are $E$ and $F$ weakly bisimilar?
- **Assume both** $T_E$ and $T_F$ are finite.
- **We consider the labelled transition system** $(S, \delta)$, where $S = S_E \cup S_F$ and $\delta = \delta_E \cup \delta_F$. 
The weak bisimilarity problem

- **Given:** two processes $E$ and $F$.
- **Decide:** is $E \approx F$? i.e., are $E$ and $F$ weakly bisimilar?
- **Assume both** $T_E$ and $T_F$ **are finite.**
- **We consider the labelled transition system** $(S, \delta)$, **where** $S = S_E \cup S_F$ **and** $\delta = \delta_E \cup \delta_F$.
- **All relations we use are subsets of** $S \times S$ **where** $S$ **is finite.**
Main idea

- The definition of weak bisimilarity is very similar to that of strong bisimilarity:

  replace \( \Rightarrow \) by \( \rightarrow \) everywhere.

- It follows: E and F are weakly bisimilar if and only if they are strongly bisimilar "with respect to the transition system \((S, \hat{\delta})\)" obtained by replacing \( \Rightarrow \) through \( \rightarrow \) in the transition system \((S, \delta)\).

- Scheme of the algorithm:
  - Compute \((S, \hat{\delta})\) such that for every action \(a\) (including \(\tau\)) and every pair of states \(s, s' \in S\), \(s a \xrightarrow{} s'\) in \((S, \hat{\delta})\) if and only if \(s a \Rightarrow s'\) in \((S, \delta)\).
  - Check if \(E \sim F\) "with respect to the transition system \((S, \hat{\delta})\)."
Main idea

- The definition of weak bisimilarity is very similar to that of strong bisimilarity:
  
  \[ \text{replace } \Rightarrow \text{ by } \rightarrow \text{ everywhere.} \]

- It follows:

  \[ E \text{ and } F \text{ are weakly bisimilar if and only if they are strongly bisimilar “with respect to the transition system } (S, \hat{\delta}) \text{” obtained by replacing } \Rightarrow \text{ through } \rightarrow \text{ in the transition system } (S, \delta). \]
Main idea

- The definition of weak bisimilarity is very similar to that of strong bisimilarity:

  replace $\Rightarrow$ by $\rightarrow$ everywhere.

- It follows:

  $E$ and $F$ are weakly bisimilar if and only if they are strongly bisimilar “with respect to the transition system $(S, \hat{\delta})$” obtained by replacing $\Rightarrow$ through $\rightarrow$ in the transition system $(S, \delta)$.

- Scheme of the algorithm:
Main idea

- The definition of weak bisimilarity is very similar to that of strong bisimilarity:
  
  replace $\Rightarrow$ by $\rightarrow$ everywhere.

- It follows:
  
  $E$ and $F$ are weakly bisimilar if and only if they are strongly bisimilar “with respect to the transition system $(S, \hat{\delta})$” obtained by replacing $\Rightarrow$ through $\rightarrow$ in the transition system $(S, \delta)$.

- Scheme of the algorithm:
  
  - Compute $(S, \hat{\delta})$ such that for every action $a$ (including $\tau$) and every pair of states $s, s' \in S$, $s \xrightarrow{a} s'$ in $(S, \hat{\delta})$ if and only if $s \xrightarrow{a} s'$ in $(S, \delta)$.
Main idea

- The definition of weak bisimilarity is very similar to that of strong bisimilarity:

  replace $\Rightarrow$ by $\rightarrow$ everywhere.

- It follows:

  \[ E \text{ and } F \text{ are weakly bisimilar if and only if they are strongly bisimilar "with respect to the transition system } (S, \hat{\delta}) \text{" obtained by replacing } \Rightarrow \text{ through } \rightarrow \text{ in the transition system } (S, \delta). \]

- Scheme of the algorithm:
  - Compute \((S, \hat{\delta})\) such that for every action \(a\) (including \(\tau\)) and every pair of states \(s, s' \in S\), \(s \xrightarrow{a} s'\) in \((S, \hat{\delta})\) if and only if \(s \xrightarrow{a} s'\) in \((S, \delta)\).
  - Check if \(E \sim F\) “with respect to the transition system \((S, \hat{\delta})\).”
Computing \((S, \hat{\delta})\)

We consider an abstract algorithm first

Input: \((S, \delta)\)
Output: \((S, \hat{\delta})\)

Initialize \(\hat{\delta} := \delta \cup \{(s, \tau, s) \mid s \in S\} \);  
Iterate: For every action \(a\) and \(s, s', s'' \in S\)  
\>
If \((s, a, s') \in \hat{\delta}\) and \((s', \tau, s'') \in \hat{\delta}\) or \((s, \tau, s') \in \hat{\delta}\) and \((s', a, s'') \in \hat{\delta}\)  
then add \((s, a, s'')\) to \(\hat{\delta}\)  
until a fixpoint is reached;  
return \((S, \hat{\delta})\)
Correctness and complexity

- Correctness: Exercise
Correctness and complexity

- Correctness: Exercise
- Complexity:

\[ O(|S|^2 \cdot |A|) \text{ iterations} \]

\[ O(|S|^3 \cdot |A|) \text{ time per iteration} \]

Overall time complexity:

\[ O(|S|^5 \cdot |A|^2) \]

Space complexity:

\[ O(|S|^2 \cdot |A|) \]
Correctness and complexity

- Correctness: Exercise
- Complexity:
  - \( O(|S|^2 \cdot |A|) \) iterations
Correctness and complexity

- **Correctness:** Exercise
- **Complexity:**
  - $O(|S|^2 \cdot |A|)$ iterations
  - $O(|S|^3 \cdot |A|)$ time per iteration
Correctness and complexity

- **Correctness:** Exercise
- **Complexity:**
  - $O(|S|^2 \cdot |A|)$ iterations
  - $O(|S|^3 \cdot |A|)$ time per iteration
- **Overall time complexity:** $O(|S|^5 \cdot |A|^2)$
Correctness and complexity

- Correctness: Exercise
- Complexity:
  - \( O(|S|^2 \cdot |A|) \) iterations
  - \( O(|S|^3 \cdot |A|) \) time per iteration
- Overall time complexity: \( O(|S|^5 \cdot |A|^2) \)
- Space complexity: \( O(|S|^2 \cdot |A|) \)
A better algorithm

Input: \((S, \delta)\)    Output: \((S, \hat{\delta})\)
1. Initialize \(\hat{\delta} := \emptyset\);
2. Initialize \(\rho := \delta \cup \{(s, \tau, s) \mid s \in S\}\);
3. while \(\rho \neq \emptyset\) do
   4. remove \(t = (s, a, s')\) from \(\rho\);
   5. if \(t \notin \hat{\delta}\) then
      6. add \(t\) to \(\hat{\delta}\);
   7. for all \(s''\) such that \((s'', \tau, s)\) \(\in \hat{\delta}\)
      8. if \((s'', a, s')\) \(\notin \rho\)
         9. then add \((s'', a, s')\) to \(\rho\);
   10. for all \(s''\) such that \((s', \tau, s'')\) \(\in \hat{\delta}\)
      11. if \((s, a, s'')\) \(\notin \rho\)
         12. then add \((s, a, s'')\) to \(\rho\);
13. return \((S, \hat{\delta})\)
Correctness (w.r.t = with respect to)

- **Termination.** Every iteration removes an element from $\rho$, but only finitely many add elements to it (because of line 5).
Correctness (w.r.t = with respect to)

- **Termination.** Every iteration removes an element from $\rho$, but only finitely many add elements to it (because of line 5).
- If $(s, a, s') \in \hat{\delta}$ after termination, then $s \xrightarrow{a} s'$ w.r.t $\delta$. (Easy)
Correctness (w.r.t = with respect to)

▶ Termination. Every iteration removes an element from $\rho$, but only finitely many add elements to it (because of line 5).

▶ If $(s, a, s') \in \hat{\delta}$ after termination, then $s \xrightarrow{a} s'$ w.r.t $\delta$. (Easy)

▶ If $s \xrightarrow{a} s'$ w.r.t $\delta$, then $(s, a, s') \in \hat{\delta}$ after termination.

Proof: By induction on the length $n$ of the shortest sequence showing $s \xrightarrow{a} s'$. The base $n = 0$ is easy (this is the case $s = s'$ and $a = \tau$). For $n > 0$, we consider two cases:
Correctness (w.r.t = with respect to)

- **Termination.** Every iteration removes an element from $\rho$, but only finitely many add elements to it (because of line 5).

- If $(s, a, s') \in \hat{\delta}$ after termination, then $s \xrightarrow{a} s'$ w.r.t $\delta$. (Easy)

- If $s \xrightarrow{a} s'$ w.r.t $\delta$, then $(s, a, s') \in \hat{\delta}$ after termination.

  **Proof:** By induction on the length $n$ of the shortest sequence showing $s \xrightarrow{a} s'$. The base $n = 0$ is easy (this is the case $s = s'$ and $a = \tau$). For $n > 0$, we consider two cases:

  - There is a $s''$ such that $(s, \tau, s'') \in \delta$ and $s'' \xrightarrow{a} s'$ with respect to $\delta$. Since the shortest sequence showing $s'' \xrightarrow{a} s'$ has length $n - 1$, by induction hypothesis $(s'', a, s')$ is eventually added to $\hat{\delta}$. Since any element that is moved to $\delta$ comes from $\rho$, $(s'', a, s')$ must be eventually added to $\rho$. By lines 7-9, $(s, a, s')$ is also eventually added to $\rho$, and so to $\hat{\delta}$. 


Correctness (w.r.t = with respect to)

- **Termination.** Every iteration removes an element from $\rho$, but only finitely many add elements to it (because of line 5).

- If $(s, a, s') \in \hat{\delta}$ after termination, then $s \xrightarrow{a} s'$ w.r.t $\delta$. (**Easy**)

- If $s \xrightarrow{a} s'$ w.r.t $\delta$, then $(s, a, s') \in \hat{\delta}$ after termination.

**Proof:** By induction on the length $n$ of the shortest sequence showing $s \xrightarrow{a} s'$. The base $n = 0$ is easy (this is the case $s = s'$ and $a = \tau$). For $n > 0$, we consider two cases:

- There is a $s''$ such that $(s, \tau, s'') \in \delta$ and $s'' \xrightarrow{a} s'$ with respect to $\delta$. Since the shortest sequence showing $s'' \xrightarrow{a} s'$ has length $n - 1$, by induction hypothesis $(s'', a, s')$ is eventually added to $\hat{\delta}$. Since any element that is moved to $\delta$ comes from $\rho$, $(s'', a, s')$ must be eventually added to $\rho$. By lines 7-9, $(s, a, s')$ is also eventually added to $\rho$, and so to $\hat{\delta}$.

- There is $s''$ such that $(s'', \tau, s') \in \delta$ and $s \xrightarrow{a} s''$ with respect to $\delta$. Analogous argument to the previous case, this time using lines lines 10-12.
Time and space complexity

Time complexity:

1. Line 6 is executed $O(|S|^2 \cdot |A|)$ times. No transition can be added to $\hat{\delta}$ twice because of line 5. Since there are at most $|S| \cdot |A| \cdot |S|$ transitions, the bound follows.

2. Lines 8 and 11 are executed $O(|S|^3 \cdot |A|)$ times. They are executed at most once for each combination $s, s', s'', a$, because no element is added to $\hat{\delta}$ twice.

3. Line 4 is executed $O(|S|^3 \cdot |A|)$ times. By 2., $O(|S|^3 \cdot |A|)$ elements are added to $\rho$ during the execution of the algorithm, and so $O(|S|^3 \cdot |A|)$ elements are have been removed from it after termination.

4. Lines 1, 2, and 13 take together $O(|S|^2 \cdot |A|)$ time.

5. The overall time complexity is $O(|S|^3 \cdot |A|)$.

Space complexity: since $\rho$ and $\hat{\delta}$ do not contain duplicates, they require $O(|S|^2 \cdot |A|)$ space.