Widening Automata
Verification of assertions
Forward fixpoint computation

1. Set of initial states $I$
2. Postcondition function $Post()$
3. Error states
Verification of assertions
Backward fixpoint computation

1. Set of initial states I
2. Precondition function Pre()
3. Error states E
Fixpoints may not converge

- For infinite state systems fixpoint computations may
  - not converge at all
  - require a large number of iterations

- Widening is a **approximation** technique that helps a fixpoint computation converge
A widening operator

- Idea: Instead of computing a sequence of automata $A_1, A_2, \ldots$ where $A_{i+1} = A_i \cup \text{post}(A_i)$, compute $A'_1, A'_2, \ldots$ where
  $A_{i+1}' = A'_i \bigtriangleup (A'_i \cup \text{post}(A'_i))$
- By definition $A \cup B \subseteq A \bigtriangleup B$
- The goal is to find a widening operator $\bigtriangleup$ such that:
  1. The sequence $A'_1, A'_2, \ldots$ converges
  2. It converges fast
  3. The computed fixpoint is as close as possible to the exact set of reachable states
Widening Automata

- Given automata $A$ and $A'$ we want to compute $A \vee A'$.
- We say that states $k$ and $k'$ are equivalent ($k \equiv k'$) if either
  - $k$ and $k'$ can be reached from either initial state with the same string (unless $k$ or $k'$ is a sink state)
  - or, the languages accepted from $k$ and $k'$ are equal
  - or, for some state $k''$, $k \equiv k''$ and $k' \equiv k''$
- The states of $A \vee A'$ are the equivalence classes of $\equiv$. 
Example

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- State 0: Transitions on 0 and 1, leading to state 1.
- State 1: Transitions on 0, 1, and X, leading to states 0, 1, and X.
- State 2: Transitions on 0 and 1, leading to state 3.
- State 3: Transitions on 0 and 1, leading to state X.
Example
## Example

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![State Transition Diagram](image-url)
Example
An exactness result (some definitions)

- An automaton \((Q, \Sigma, \delta, q_0, F)\) is called state-disjoint if for all \(q_i \neq q_j \in Q\), \(L(q_i) \cap L(q_j) = \emptyset\).

- An automaton \((Q_1, \Sigma, \delta_1, q_{01}, F_1)\) is called weakly equivalent to \((Q_2, \Sigma, \delta_2, q_{02}, F_2)\) iff there exists \(f: Q_1 \rightarrow Q_2\), such that:

  - \(f(q_{01}) = q_{02}\)
  - \(f(\delta_1(q, \sigma)) = \delta_2(f(q), \sigma)\) for all \(q \in Q_1\) and \(\sigma \in \Sigma\)
  - \(f(q) \in F_2\) for all \(q \in F_1\)
An exactness result

- If
  - a least fixpoint is represented by a state-disjoint automaton $A_\infty$
  - and, the first automaton $A_s$ in the approximate sequence is weakly equivalent to $A_\infty$

- then
  - the approximate sequence converges to the exact least fixpoint