COMP2111 Week 7
Term 1, 2019
State machines
Summary

- Motivation
- Definitions
- The invariant principle
- Partial correctness and termination
- Input and output
- Finite automata
Summary

- Motivation
- Definitions
- The invariant principle
- Partial correctness and termination
- Input and output
- Finite automata
Motivation: Models of computation

State machines model step-by-step processes:
- Set of “states”, possibly including a designated “start state”
- For each state, a set of actions detailing how to move (transition) to other states

Example

The semantics of a program in $L$:
- States: functions from variables to numerical values
- Transitions: defined by the program
Motivation: Models of computation

State machines model step-by-step processes:
- Set of “states”, possibly including a designated “start state”
- For each state, a set of actions detailing how to move (transition) to other states

Example

A chess solving engine
- States: Board positions
- Transitions: Legal moves
Motivation: Models of computation

State machines model step-by-step processes:

- Set of “states”, possibly including a designated “start state”
- For each state, a set of actions detailing how to move (transition) to other states

Example

“Stateful” communication protocols: e.g. SMTP

- States: Stages of communication
- Transitions: Determined by commands given (e.g. HELO, DATA, etc)
Motivation: Models of computation

State machines model step-by-step processes:
- Set of “states”, possibly including a designated “start state”
- For each state, a set of actions detailing how to move (transition) to other states

Example

A bounded counter that counts from 0 to 99 and overflows at 100:
Motivation: Models of computation

State machines model step-by-step processes:
- Set of “states”, possibly including a designated “start state”
- For each state, a set of actions detailing how to move (transition) to other states

Example

A robot that moves diagonally

States: Locations
Transitions: Moves
Motivation: Models of computation

State machines model step-by-step processes:
- Set of “states”, possibly including a designated “start state”
- For each state, a set of actions detailing how to move (transition) to other states

Example

Die Hard jug problem: Given jugs of 3L and 5L, measure out exactly 4L.
- States: Defined by amount of water in each jug
- Start state: No water in both jugs
- Transitions: Pouring water (in, out, jug-to-jug)
Summary

- Motivation
- Definitions
- The invariant principle
- Partial correctness and termination
- Input and output
- Finite automata
Definitions

A transition system is a pair \((S, \rightarrow)\) where:

- \(S\) is a set (of states), and
- \(\rightarrow \subseteq S \times S\) is a (transition) relation.

If \((s, s') \in \rightarrow\) we write \(s \rightarrow s'\).

- \(S\) may have a designated start state, \(s_0 \in S\)
- \(S\) may have designated final states, \(F \subseteq S\)
- The transitions may be labelled by elements of a set \(\Lambda\):
  - \(\rightarrow \subseteq S \times \Lambda \times S\)
  - \((s, a, s') \in \rightarrow\) is written as \(s \xrightarrow{a} s'\)
- If \(\rightarrow\) is a function we say the system is deterministic, otherwise it is non-deterministic
Definitions

A transition system is a pair \((S, \rightarrow)\) where:

- \(S\) is a set (of states), and
- \(\rightarrow \subseteq S \times S\) is a (transition) relation.

If \((s, s') \in \rightarrow\) we write \(s \rightarrow s'\).

- \(S\) may have a designated start state, \(s_0 \in S\)
- \(S\) may have designated final states, \(F \subseteq S\)
- The transitions may be labelled by elements of a set \(\Lambda\):
  - \(\rightarrow \subseteq S \times \Lambda \times S\)
  - \((s, a, s') \in \rightarrow\) is written as \(s \xrightarrow{a} s'\)
- If \(\rightarrow\) is a function we say the system is deterministic, otherwise it is non-deterministic
Definitions

A **transition system** is a pair \((S, \rightarrow)\) where:

- \(S\) is a set (of **states**), and
- \(\rightarrow \subseteq S \times S\) is a (transition) relation.

If \((s, s') \in \rightarrow\) we write \(s \rightarrow s'\).

- \(S\) may have a designated **start state**, \(s_0 \in S\)
- \(S\) may have designated **final states**, \(F \subseteq S\)
- The transitions may be **labelled** by elements of a set \(\Lambda\):
  - \(\rightarrow \subseteq S \times \Lambda \times S\)
  - \((s, a, s') \in \rightarrow\) is written as \(s \stackrel{a}{\rightarrow} s'\)

- If \(\rightarrow\) is a function we say the system is deterministic, otherwise it is **non-deterministic**
A transition system is a pair \((S, \rightarrow)\) where:

- \(S\) is a set (of states), and
- \(\rightarrow \subseteq S \times S\) is a (transition) relation.

If \((s, s') \in \rightarrow\) we write \(s \rightarrow s'\).

- \(S\) may have a designated start state, \(s_0 \in S\)
- \(S\) may have designated final states, \(F \subseteq S\)
- The transitions may be labelled by elements of a set \(\Lambda\):
  - \(\rightarrow \subseteq S \times \Lambda \times S\)
  - \((s, a, s') \in \rightarrow\) is written as \(s \xrightarrow{a} s'\)
- If \(\rightarrow\) is a function we say the system is deterministic, otherwise it is non-deterministic.
Example: Bounded counter

A bounded counter that counts from 0 to 99 and overflows at 100:

- \( S = \{0, 1, \ldots, 99, \text{overflow}\} \)
- \( \{(i, i+1) : 0 \leq i < 99\} \)
- \( \rightarrow = \bigcup \{(99, \text{overflow})\} \)
- \( \bigcup \{(\text{overflow, overflow})\} \)
- \( s_0 = 0 \)
- Deterministic
Example: Diagonally moving robot

States: Locations
Transitions: Moves
Example: Diagonally moving robot

\[ S = \mathbb{Z} \times \mathbb{Z} \]

\[(x, y) \rightarrow (x \pm 1, y \pm 1)\]

Non-deterministic
Example: Diagonally moving robot

\[ S = \mathbb{Z} \times \mathbb{Z} \]

\[ \Lambda = \{ \text{NW, NE, SW, SE} \} \]

\[ (x, y) \xrightarrow{\text{NW}} (x - 1, y + 1) \]

\[ (x, y) \xrightarrow{\text{NE}} (x + 1, y + 1) \]

\[ (x, y) \xrightarrow{\text{SW}} (x - 1, y - 1) \]

\[ (x, y) \xrightarrow{\text{SE}} (x + 1, y - 1) \]

Deterministic
Example: Die Hard jug problem

Example

Given jugs of 3L and 5L, measure out exactly 4L.

- States: Defined by amount of water in each jug
- Start state: No water in both jugs
- Transitions: Pouring water (in, out, jug-to-jug)
Example: Die Hard jug problem

Example

Given jugs of 3L and 5L, measure out exactly 4L.

- \( S = \{(i,j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\} \)
- \( s_0 = (0, 0) \)
- \( \rightarrow \) given by
  - \( (i,j) \rightarrow (0,j) \) \[empty 5L jug\]
  - \( (i,j) \rightarrow (i,0) \) \[empty 3L jug\]
  - \( (i,j) \rightarrow (5,j) \) \[fill 5L jug\]
  - \( (i,j) \rightarrow (i,3) \) \[fill 3L jug\]
  - \( (i,j) \rightarrow (i+j,0) \) if \( i+j \leq 5 \) \[empty 3L jug into 5L jug\]
  - \( (i,j) \rightarrow (0,i+j) \) if \( i+j \leq 3 \) \[empty 5L jug into 3L jug\]
  - \( (i,j) \rightarrow (5,j-5+i) \) if \( i+j \geq 5 \) \[fill 5L jug from 3L jug\]
  - \( (i,j) \rightarrow (i-3+j,3) \) if \( i+j \geq 3 \) \[fill 3L jug from 5L jug\]
Example: Die Hard jug problem

Example

Given jugs of 3L and 5L, measure out exactly 4L.

\[ S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\} \]

\[ s_0 = (0, 0) \]

→ given by

- \( (i, j) \rightarrow (0, j) \) \ [empty 5L jug]
- \( (i, j) \rightarrow (i, 0) \) \ [empty 3L jug]
- \( (i, j) \rightarrow (5, j) \) \ [fill 5L jug]
- \( (i, j) \rightarrow (i, 3) \) \ [fill 3L jug]
- \( (i, j) \rightarrow (i + j, 0) \) if \( i + j \leq 5 \) \ [empty 3L jug into 5L jug]
- \( (i, j) \rightarrow (0, i + j) \) if \( i + j \leq 3 \) \ [empty 5L jug into 3L jug]
- \( (i, j) \rightarrow (5, j - 5 + i) \) if \( i + j \geq 5 \) \ [fill 5L jug from 3L jug]
- \( (i, j) \rightarrow (i - 3 + j, 3) \) if \( i + j \geq 3 \) \ [fill 3L jug from 5L jug]
Example: Die Hard jug problem

Example

Given jugs of 3L and 5L, measure out exactly 4L.

- \( S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\} \)
- \( s_0 = (0, 0) \)
- \( \rightarrow \) given by
  - \((i, j) \rightarrow (0, j)\)  \[empty 5L jug\]
  - \((i, j) \rightarrow (i, 0)\)  \[empty 3L jug\]
  - \((i, j) \rightarrow (5, j)\) \[fill 5L jug\]
  - \((i, j) \rightarrow (i, 3)\) \[fill 3L jug\]
  - \((i, j) \rightarrow (i + j, 0)\) if \(i + j \leq 5\) \[empty 3L jug into 5L jug\]
  - \((i, j) \rightarrow (0, i + j)\) if \(i + j \leq 3\) \[empty 5L jug into 3L jug\]
  - \((i, j) \rightarrow (5, j - 5 + i)\) if \(i + j \geq 5\) \[fill 5L jug from 3L jug\]
  - \((i, j) \rightarrow (i - 3 + j, 3)\) if \(i + j \geq 3\) \[fill 3L jug from 5L jug\]
Example: Die Hard jug problem

Given jugs of 3L and 5L, measure out exactly 4L.

\[ S = \{(i,j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\} \]

\[ s_0 = (0,0) \]

\[ \rightarrow \text{ given by} \]

- \( (i,j) \rightarrow (0,j) \) \[ \text{[empty 5L jug]} \]
- \( (i,j) \rightarrow (i,0) \) \[ \text{[empty 3L jug]} \]
- \( (i,j) \rightarrow (5,j) \) \[ \text{[fill 5L jug]} \]
- \( (i,j) \rightarrow (i,3) \) \[ \text{[fill 3L jug]} \]
- \( (i,j) \rightarrow (i+j,0) \) if \( i+j \leq 5 \) \[ \text{[empty 3L jug into 5L jug]} \]
- \( (i,j) \rightarrow (0,i+j) \) if \( i+j \leq 3 \) \[ \text{[empty 5L jug into 3L jug]} \]
- \( (i,j) \rightarrow (5,j-5+i) \) if \( i+j \geq 5 \) \[ \text{[fill 5L jug from 3L jug]} \]
- \( (i,j) \rightarrow (i-3+j,3) \) if \( i+j \geq 3 \) \[ \text{[fill 3L jug from 5L jug]} \]
Given jugs of 3L and 5L, measure out exactly 4L.

\[ S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\} \]

\[ s_0 = (0, 0) \]

\[ \rightarrow \text{ given by} \]

- \((i, j) \rightarrow (0, j)\) \[\text{[empty 5L jug]}\]
- \((i, j) \rightarrow (i, 0)\) \[\text{[empty 3L jug]}\]
- \((i, j) \rightarrow (5, j)\) \[\text{[fill 5L jug]}\]
- \((i, j) \rightarrow (i, 3)\) \[\text{[fill 3L jug]}\]
- \((i, j) \rightarrow (i + j, 0)\) if \(i + j \leq 5\) \[\text{[empty 3L jug into 5L jug]}\]
- \((i, j) \rightarrow (0, i + j)\) if \(i + j \leq 3\) \[\text{[empty 5L jug into 3L jug]}\]
- \((i, j) \rightarrow (5, j - 5 + i)\) if \(i + j \geq 5\) \[\text{[fill 5L jug from 3L jug]}\]
- \((i, j) \rightarrow (i - 3 + j, 3)\) if \(i + j \geq 3\) \[\text{[fill 3L jug from 5L jug]}\]
Example: Die Hard jug problem

Example

Given jugs of 3L and 5L, measure out exactly 4L.

• $S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5$ and $0 \leq j \leq 3\}$
• $s_0 = (0, 0)$
• $\rightarrow$ given by
  • $(i, j) \rightarrow (0, j)$  [empty 5L jug]
  • $(i, j) \rightarrow (i, 0)$  [empty 3L jug]
  • $(i, j) \rightarrow (5, j)$  [fill 5L jug]
  • $(i, j) \rightarrow (i, 3)$  [fill 3L jug]
  • $(i, j) \rightarrow (i + j, 0)$ if $i + j \leq 5$  [empty 3L jug into 5L jug]
  • $(i, j) \rightarrow (0, i + j)$ if $i + j \leq 3$  [empty 5L jug into 3L jug]
  • $(i, j) \rightarrow (5, j - 5 + i))$ if $i + j \geq 5$  [fill 5L jug from 3L jug]
  • $(i, j) \rightarrow (i - 3 + j, 3)$ if $i + j \geq 3$  [fill 3L jug from 5L jug]
Runs and reachability

Given a transition system $(S, \rightarrow)$ and states $s, s' \in S$,

- a run from $s$ is a (possibly infinite) sequence $s_1, s_2, \ldots$ such that $s = s_1$ and $s_i \rightarrow s_{i+1}$ for all $i \geq 1$.

- we say $s'$ is reachable from $s$, written $s \rightarrow^* s'$, if $(s, s')$ is in the transitive closure of $\rightarrow$.

NB

$s'$ is reachable from $s$ if there is a run from $s$ which contains $s'$. 
Runs and reachability

Given a transition system \((S, \rightarrow)\) and states \(s, s' \in S\),

- a **run** from \(s\) is a (possibly infinite) sequence \(s_1, s_2, \ldots\) such that \(s = s_1\) and \(s_i \rightarrow s_{i+1}\) for all \(i \geq 1\).

- we say \(s'\) is **reachable** from \(s\), written \(s \rightarrow^* s'\), if \((s, s')\) is in the transitive closure of \(\rightarrow\).

**NB**

\(s'\) is reachable from \(s\) if there is a run from \(s\) which contains \(s'\).
Safety and Liveness

Common problem (Safety)
Will a transition system always avoid a particular state or states?
Equivalently, can a transition system reach a particular state or states?

Common problem (Liveness)
Will a transition system always reach a particular state or states?
Equivalently, can a transition system avoid a particular state or states?
Safety and Liveness

**Common problem (Safety)**
Will a transition system always avoid a particular state or states? Equivalently, can a transition system reach a particular state or states?

**Common problem (Liveness)**
Will a transition system always reach a particular state or states? Equivalently, can a transition system avoid a particular state or states?
Safety and Liveness

Common problem (Safety)
Will a transition system always avoid a particular state or states? Equivalently, can a transition system reach a particular state or states?

Common problem (Liveness)
Will a transition system always reach a particular state or states? Equivalently, can a transition system avoid a particular state or states?
Reachability example: Die Hard jug problem

Example

Given jugs of 3L and 5L, measure out exactly 4L.

- States: $S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5$ and $0 \leq j \leq 3\}$
- Transition relation: $(i, j) \rightarrow (0, j)$ etc.

Is $(4, 0)$ reachable from $(0, 0)$?

Yes:

$(0, 0) \rightarrow (0, 3) \rightarrow (3, 0)$

$(0, 1) \leftarrow (5, 1) \leftarrow (3, 3)$

$(1, 0) \rightarrow (1, 3) \rightarrow (4, 0)$
Reachability example: Die Hard jug problem

**Example**

Given jugs of 3L and 5L, measure out exactly 4L.

- **States:** \( S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\} \)
- **Transition relation:** \((i, j) \rightarrow (0, j)\) etc.

Is \((4, 0)\) reachable from \((0, 0)\)?

Yes:

\[
(0, 0) \rightarrow (0, 3) \rightarrow (3, 0) \\
(0, 1) \leftarrow (5, 1) \leftarrow (3, 3) \\
(1, 0) \rightarrow (1, 3) \rightarrow (4, 0)
\]
Safety example: Diagonally moving robot

Example

Starting at \((0, 0)\)
Can the robot get to \((0, 1)\)?
Safety example: Diagonally moving robot

Example

Starting at \((0, 0)\)

Can the robot get to \((0, 1)\)?
Safety example: Diagonally moving robot

Example

Starting at \((0, 0)\)
Can the robot get to \((0, 1)\)? No
Safety example: Diagonally moving robot

Example

Starting at (0, 0)
Can the robot get to (0, 1)? No

isBlue((m, n)) := 2|(m + n)
Safety example: Diagonally moving robot

Example

Starting at (0, 0)
Can the robot get to (0, 1)? No

isBlue((m, n)) := 2|(|m + n)
if isBlue(s) and s → s'
then isBlue(s')
Safety example: Diagonally moving robot

Example

Starting at (0, 0)

Can the robot get to (0, 1)? No

isBlue((m, n)) := 2|(m + n)

if isBlue(s) and s → s' then isBlue(s')

isBlue((0, 0)) and ¬isBlue((0, 1))
Summary

- Motivation
- Definitions
- The invariant principle
- Partial correctness and termination
- Input and output
- Finite automata
A preserved invariant of a transition system is a unary predicate $\varphi$ on states such that if $\varphi(s)$ holds and $s \rightarrow s'$ then $\varphi(s')$ holds.

**Invariant principle**

If a preserved invariant holds at a state $s$, then it holds for all states reachable from $s$. 

**Proof:**
The invariant principle

A preserved invariant of a transition system is a unary predicate \( \varphi \) on states such that if \( \varphi(s) \) holds and \( s \rightarrow s' \) then \( \varphi(s') \) holds.

**Invariant principle**

If a preserved invariant holds at a state \( s \), then it holds for all states reachable from \( s \).

Proof:
Example

Given jugs of 3L and 6L, measure out exactly 4L.

- States: \( S = \{(i,j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 6 \text{ and } 0 \leq j \leq 3\} \)
- Transition relation: \((i,j) \rightarrow (0,j)\) etc.

Is \((4,0)\) reachable from \((0,0)\)?

No. Consider \( \varphi((i,j)) = (3|i) \land (3|j) \).
Example

Given jugs of 3L and 6L, measure out exactly 4L.

- States: \( S = \{(i,j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 6 \text{ and } 0 \leq j \leq 3\} \)
- Transition relation: \((i,j) \rightarrow (0,j)\) etc.

Is \((4,0)\) reachable from \((0,0)\)?

No. Consider \(\varphi((i,j)) = (3|i) \land (3|j)\).
Summary

- Motivation
- Definitions
- The invariant principle
- Partial correctness and termination
- Input and output
- Finite automata
Partial correctness

Let \((S, \rightarrow, s_0, F)\) be a transition system with start state \(s_0\) and final states \(F\) and a \(\varphi\) be a unary predicate on \(S\). We say the system is **partially correct for** \(\varphi\) if \(\varphi(s')\) holds for all states \(s' \in F\) that are reachable from \(s_0\).

**NB**

*Partial correctness does not guarantee a transition system will reach a final state.*
Partial correctness example: Fast exponentiation

Example
Consider the following program in $\mathcal{L}$:

\[
\begin{align*}
x &:= m; \\
y &:= n; \\
r &:= 1; \\
\textbf{while } & y > 0 \textbf{ do} \\
\quad & \text{if } 2|y \text{ then} \\
\quad & \quad y := y/2 \\
\quad & \text{else} \\
\quad & \quad y := (y - 1)/2; \\
\quad & \quad r := r \times x \\
\quad & \textbf{fi;} \\
x &:= x \times x \\
\textbf{od}
\end{align*}
\]
Partial correctness example: Fast exponentiation

Example

- States: Functions from \( \{m, n, x, y, r\} \) to \( \mathbb{N} \)
- Transitions: Effect of each line of code:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd
- Start state: \((m, n, 1)\)
- Final states: \(\{(x, 0, r) : x, r \in \mathbb{N}\}\)

Goal: Show partial correctness for \(\varphi((x, y, r)) := (r = m^n)\)

Show \(\psi((x, y, r)) := (rx^y = m^n)\) is a preserved invariant...

How can we show total correctness?
Partial correctness example: Fast exponentiation

Example

- States: \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- Transitions: Effect of each line of code:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd
- Start state: \((m, n, 1)\)
- Final states: \(\{(x, 0, r) : x, r \in \mathbb{N}\}\)

Goal: Show partial correctness for \(\varphi((x, y, r)) := (r = m^n)\)

Show \(\psi((x, y, r)) := (rx^y = m^n)\) is a preserved invariant...

How can we show total correctness?
Partial correctness example: Fast exponentiation

Example

- States: \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- Transitions: Effect of each iteration of while loop:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd
- Start state: \((m, n, 1)\)
- Final states: \(\{(x, 0, r) : x, r \in \mathbb{N}\}\)

Goal: Show partial correctness for \(\varphi((x, y, r)) := (r = m^n)\)

Show \(\psi((x, y, r)) := (rx^y = m^n)\) is a preserved invariant...

How can we show total correctness?
Partial correctness example: Fast exponentiation

Example

- States: \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- Transitions: Effect of each iteration of while loop:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd
- Start state: \((m, n, 1)\)
- Final states: \(\{(x, 0, r) : x, r \in \mathbb{N}\}\)

Goal: Show partial correctness for \(\varphi((x, y, r)) := (r = m^n)\)

Show \(\psi((x, y, r)) := (rx^y = m^n)\) is a preserved invariant...

How can we show total correctness?
Partial correctness example: Fast exponentiation

Example

- States: \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- Transitions: Effect of each iteration of while loop:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd
- Start state: \((m, n, 1)\)
- Final states: \(\{(x, 0, r) : x, r \in \mathbb{N}\}\)

Goal: Show partial correctness for \(\varphi((x, y, r)) := (r = m^n)\)

Show \(\psi((x, y, r)) := (rx^y = m^n)\) is a preserved invariant...

How can we show total correctness?
Partial correctness example: Fast exponentiation

Example

- States: \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- Transitions: Effect of each iteration of while loop:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd
- Start state: \((m, n, 1)\)
- Final states: \(\{(x, 0, r) : x, r \in \mathbb{N}\}\)

Goal: Show partial correctness for \(\varphi((x, y, r)) := (r = m^n)\)

Show \(\psi((x, y, r)) := (rx^y = m^n)\) is a preserved invariant...

How can we show total correctness?
Partial correctness example: Fast exponentiation

Example

- States: \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- Transitions: Effect of each iteration of while loop:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd
- Start state: \((m, n, 1)\)
- Final states: \{\((x, 0, r) : x, r \in \mathbb{N}\}\)

Goal: Show partial correctness for \(\varphi((x, y, r)) := (r = m^n)\)

Show \(\psi((x, y, r)) := (rx^y = m^n)\) is a preserved invariant...

How can we show total correctness?
Partial correctness example: Fast exponentiation

Example

- States: \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- Transitions: Effect of each iteration of while loop:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd
- Start state: \((m, n, 1)\)
- Final states: \(\{(x, 0, r) : x, r \in \mathbb{N}\}\)

Goal: Show partial correctness for \(\varphi((x, y, r)) := (r = m^n)\)

Show \(\psi((x, y, r)) := (rx^y = m^n)\) is a preserved invariant...

How can we show total correctness?
A transition system \((S, \rightarrow)\) terminates from a state \(s \in S\) if there is an \(N \in \mathbb{N}\) such that all runs from \(s\) have length at most \(N\).

A transition system is \textbf{totally correct for a unary predicate} \(\varphi\), if it terminates (from \(s_0\)) and \(\varphi\) holds in the last state of every run.
Derived variables

In a transition system \((S, \rightarrow)\), a **derived variable** is a function \(f : S \rightarrow \mathbb{R}\).

A derived variable is **strictly decreasing** if \(s \rightarrow s'\) implies \(f(s') < f(s)\).

**Theorem**

If \(f\) is an \(\mathbb{N}\)-valued, strictly decreasing derived variable, then the length of any run from \(s\) is at most \(f(s)\).
Derived variables

In a transition system \((S, \rightarrow)\), a **derived variable** is a function \(f : S \rightarrow \mathbb{R}\).

A derived variable is **strictly decreasing** if \(s \rightarrow s'\) implies \(f(s') < f(s)\).

**Theorem**

If \(f\) is an \(\mathbb{N}\)-valued, strictly decreasing derived variable, then the length of any run from \(s\) is at most \(f(s)\).
Example

- **States:** \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- **Transitions:** Effect of each iteration of while loop:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd

**Derived variable:** \(f((x, y, r)) = y\)
Termination example: Fast exponentiation

Example

- States: \((x, y, r) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}\)
- Transitions: Effect of each iteration of while loop:
  - \((x, y, r) \rightarrow (x^2, y/2, r)\) if \(y\) is even
  - \((x, y, r) \rightarrow (x^2, (y - 1)/2, rx)\) if \(y\) is odd

Derived variable: \(f((x, y, r)) = y\)
Summary

- Motivation
- Definitions
- The invariant principle
- Partial correctness and termination
- **Input and output**
- Finite automata
Interaction with the environment

We can model the system interacting with an external entity via inputs ($\Sigma$) and outputs ($\Gamma$) by using labelled transitions:

$$\rightarrow \subseteq S \times \Lambda \times S \text{ where } \Lambda = \Sigma \times \Gamma$$

Two main categories of input/output transition systems:

**Acceptors:** Accept/reject a sequence of inputs (Relations)

**Transducers:** Take a sequence of inputs and produce a sequence of outputs (Functions)
Interaction with the environment

We can model the system interacting with an external entity via inputs ($\Sigma$) and outputs ($\Gamma$) by using labelled transitions:

$$\rightarrow \subseteq S \times \Lambda \times S \text{ where } \Lambda = \Sigma \times \Gamma$$

Two main categories of input/output transition systems:

**Acceptors:** Accept/reject a sequence of inputs (Relations)

**Transducers:** Take a sequence of inputs and produce a sequence of outputs (Functions)
Acceptor example: Diagonally moving robot

Example

\[ S = \mathbb{Z} \times \mathbb{Z} \]

\[ s_0 = (0, 0) \]

\[ (x, y) \xrightarrow{NW} (x - 1, y + 1) \]

\[ (x, y) \xrightarrow{NE} (x + 1, y + 1) \]

\[ (x, y) \xrightarrow{SW} (x - 1, y - 1) \]

\[ (x, y) \xrightarrow{SE} (x + 1, y - 1) \]

Accept if \((2, 2)\) reached
Acceptor example: Diagonally moving robot

Example

$S = \mathbb{Z} \times \mathbb{Z}$

$s_0 = (0, 0)$

$(x, y) \xrightarrow{NW} (x - 1, y + 1)$
$(x, y) \xrightarrow{NE} (x + 1, y + 1)$
$(x, y) \xrightarrow{SW} (x - 1, y - 1)$
$(x, y) \xrightarrow{SE} (x + 1, y - 1)$

Accept if $(2, 2)$ reached

Accepted sequences:
$NE, NE$
Acceptor example: Diagonally moving robot

Example

\[ S = \mathbb{Z} \times \mathbb{Z} \]

\[ s_0 = (0, 0) \]

\[ (x, y) \xrightarrow{NW} (x - 1, y + 1) \]

\[ (x, y) \xrightarrow{NE} (x + 1, y + 1) \]

\[ (x, y) \xrightarrow{SW} (x - 1, y - 1) \]

\[ (x, y) \xrightarrow{SE} (x + 1, y - 1) \]

Accept if \((2, 2)\) reached

Accepted sequences:

\[ NE, NE \]

\[ NE, SE, NE, NW \]
Acceptor example: Diagonally moving robot

**Example**

\[ S = \mathbb{Z} \times \mathbb{Z} \]

\[ s_0 = (0, 0) \]

\[ (x, y) \xrightarrow{NW} (x - 1, y + 1) \]

\[ (x, y) \xrightarrow{NE} (x + 1, y + 1) \]

\[ (x, y) \xrightarrow{SW} (x - 1, y - 1) \]

\[ (x, y) \xrightarrow{SE} (x + 1, y - 1) \]

Accept if \((2, 2)\) reached

Accepted sequences:

\[ NE, NE \]

\[ NE, SE, NE, NW \]

\[ NE, NE, NE, SW \ldots \]
Transducer example: Diagonally moving robot

Example

\[ S = \mathbb{Z} \times \mathbb{Z} \]

\[ s_0 = (0, 0) \]

\[ (x, y) \xrightarrow{NW/x} (x - 1, y + 1) \]

\[ (x, y) \xrightarrow{NE/x} (x + 1, y + 1) \]

\[ (x, y) \xrightarrow{SW/x} (x - 1, y - 1) \]

\[ (x, y) \xrightarrow{SE/x} (x + 1, y - 1) \]

Input direction
Output \(x\)-coordinate
Transducer example: Diagonally moving robot

Example

\[ S = \mathbb{Z} \times \mathbb{Z} \]

\[ s_0 = (0, 0) \]

\[(x, y) \xrightarrow{NW/x} (x - 1, y + 1)\]

\[(x, y) \xrightarrow{NE/x} (x + 1, y + 1)\]

\[(x, y) \xrightarrow{SW/x} (x - 1, y - 1)\]

\[(x, y) \xrightarrow{SE/x} (x + 1, y - 1)\]

Input direction

Output \(x\)-coordinate

Input: \(NE, SE, NE, NW\)

Output: 1, 2, 3, 2
Transducer example: Diagonally moving robot

Example

\[ S = \mathbb{Z} \times \mathbb{Z} \]

\[ s_0 = (0, 0) \]

\[ (x, y) \xrightarrow{NW/y} (x - 1, y + 1) \]

\[ (x, y) \xrightarrow{NE/y} (x + 1, y + 1) \]

\[ (x, y) \xrightarrow{SW/y} (x - 1, y - 1) \]

\[ (x, y) \xrightarrow{SE/y} (x + 1, y - 1) \]

Input direction
Output \( y \)-coordinate

Input: \( NE, SE, NE, NW \)
Output: \( 1, 0, 1, 2 \)
Acceptor example: Die Hard jug problem

Example

- \( S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\} \)
- \( s_0 = (0, 0) \)
- \( \rightarrow \) given by
  - \( (i, j) \xrightarrow{E_5} (0, j) \) [empty 5L jug]
  - \( (i, j) \xrightarrow{E_3} (i, 0) \) [empty 3L jug]
  - \( (i, j) \xrightarrow{F_5} (5, j) \) [fill 5L jug]
  - \( (i, j) \xrightarrow{F_3} (i, 3) \) [fill 3L jug]
  - \( (i, j) \xrightarrow{E_35} (i + j, 0) \) if \( i + j \leq 5 \) [empty 3L jug into 5L jug]
  - \( (i, j) \xrightarrow{E_53} (0, i + j) \) if \( i + j \leq 3 \) [empty 5L jug into 3L jug]
  - \( (i, j) \xrightarrow{F_53} (5, j - 5 + i) \) if \( i + j \geq 5 \) [fill 5L jug from 3L jug]
  - \( (i, j) \xrightarrow{F_35} (i - 3 + j, 3) \) if \( i + j \geq 3 \) [fill 3L jug from 5L jug]

- Accept if \((4, 0)\) is reached: e.g. \( F_3, E_35, F_3, F_53, E_5, E_35, F_3, E_35 \)
Accept example: Die Hard jug problem

Example

- $S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\}$
- $s_0 = (0, 0)$
- $\rightarrow$ given by
  - $(i, j) \xrightarrow{E_5} (0, j)$ [empty 5L jug]
  - $(i, j) \xrightarrow{E_3} (i, 0)$ [empty 3L jug]
  - $(i, j) \xrightarrow{F_5} (5, j)$ [fill 5L jug]
  - $(i, j) \xrightarrow{F_3} (i, 3)$ [fill 3L jug]
  - $(i, j) \xrightarrow{E_{35}} (i + j, 0)$ if $i + j \leq 5$ [empty 3L jug into 5L jug]
  - $(i, j) \xrightarrow{E_{53}} (0, i + j)$ if $i + j \leq 3$ [empty 5L jug into 3L jug]
  - $(i, j) \xrightarrow{F_{53}} (5, j - 5 + i)$ if $i + j \geq 5$ [fill 5L jug from 3L jug]
  - $(i, j) \xrightarrow{F_{35}} (i - 3 + j, 3)$ if $i + j \geq 3$ [fill 3L jug from 5L jug]
- Accept if $(4, 0)$ is reached: e.g. F3, E35, F3, F53, E5, E35, F3, E35
\(\epsilon\)-transitions

It can be useful to allow the system to transition without taking input or producing output. We use the special symbol \(\epsilon\) to denote such transitions.
An acceptor is a $\Sigma \cup \{\epsilon\}$-labelled transition system $A = (S, \rightarrow, \Sigma, s_0, F)$ with a start state $s_0 \in S$ and a set of final states $F \subseteq S$.

A transducer is a $(\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\})$-labelled transition system $T = (S, \rightarrow, \Sigma, s_0, F)$ with a start state $s_0 \in S$ and a set of final states $F \subseteq S$. 
Summary

- Motivation
- Definitions
- The invariant principle
- Partial correctness and termination
- Input and output
- Finite automata
State transition systems with a finite set of states are particularly useful in Computer Science.

**Acceptors:** Finite state automata

**Transducers:** Mealy machines