COMP2111 Week 7
Term 1, 2023
State machines
Summary

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

In Assignment 1, we modelled programs as relations between initial and final states of successful executions. So this Tic-Tac-Toe program:

```c
void move(int pos, char fill) {
    if(board[pos] = "E" && (fill = "X" || fill = "O")) {
        board[pos] := fill;
    }
    else abort;
}
```

can be modelled with this relation:

\[
\{(b, b') : \exists n. \forall i. \left( n = i \rightarrow b_i = E \land b'_i \neq E \right) \land \\
\left( n \neq i \rightarrow b_i = b'_i \right)\}
\]
Motivation

Such relational modelling is useful (spoiler alert: W8-9), but doesn’t always capture everything we care about.
Motivation

Such relational modelling is useful (spoiler alert: W8-9), but doesn’t always capture everything we care about.

Possibility of failure is sometimes not captured. This:

```c
void incr() {
    if (Math.random() < .5) then abort; else
    x := x + 1;
}
```

can be modelled by this relation over $\mathbb{Z} \times \mathbb{Z}$:

$$\{(x, x') : x + 1 = x'\}$$
Motivation

Such relational modelling is useful (spoiler alert: W8-9), but doesn’t always capture everything we care about.

Possibility of failure is sometimes not captured. This:

```java
void incr() {
    if(Math.random() < .5) then abort else
        x := x + 1;
}
}
```

can also be modelled by this relation over $\mathbb{Z} \times \mathbb{Z}$:

$$\left\{ (x, x') : x + 1 = x' \right\}$$
Motivation

Such relational modelling is useful (spoiler alert: W8-9), but doesn’t always capture everything we care about.

Sometimes the final state isn’t what’s interesting.

```c
void yes() {
    while(true) print("y\n");
}
```

This program has no final states, so its relational model doesn’t say much:

```c
{}
```
Motivation

State machines model step-by-step processes with:

- A set of *states*, possibly including a designated *start state*.
- A *transition relation*, detailing how to move (transition) from one state to another.
Motivation

State machines model step-by-step processes with:

- A set of states, possibly including a designated start state.
- A transition relation, detailing how to move (transition) from one state to another.

Example

The semantics of a program:

- States: functions from variable names to values
- Transitions: execute a line of code.
Motivation

State machines model step-by-step processes with:
- A set of *states*, possibly including a designated *start state*.
- A *transition relation*, detailing how to move (transition) from one state to another.

Example

A game of noughts and crosses
- States: Board positions
- Transitions: Legal moves
Motivation

State machines model step-by-step processes with:
- A set of *states*, possibly including a designated *start state*.
- A *transition relation*, detailing how to move (transition) from one state to another.

Example

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

State machines model step-by-step processes with:
- A set of states, possibly including a designated start state.
- A transition relation, detailing how to move (transition) from one state to another.

Example

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

0 → 1 → 2 → ⋯ → 99 → overflow
Motivation

State machines model step-by-step processes with:
- A set of *states*, possibly including a designated *start state*.
- A *transition relation*, detailing how to move (transition) from one state to another.

**Example**

A robot that moves diagonally

States: Locations
Transitions: Moves
Motivation

State machines model step-by-step processes with:

- A set of states, possibly including a designated start state.
- A transition relation, detailing how to move (transition) from one state to another.

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
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 \(s \rightarrow s'\).
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, \(F \subseteq S\)
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 \(L\):
  - \(\rightarrow \subseteq S \times L \times S\)
  - \((s, a, s') \in \rightarrow\) is written as \(s \xrightarrow{a} s'\)
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 \(L\):
  - \(\rightarrow \subseteq S \times L \times S\)
  - \((s, a, s') \in \rightarrow\) is written as \(s \xrightarrow{a} s'\)
- If \(\rightarrow\) is a partial function we say that 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:

- $S = \{0, 1, \ldots, 99, \text{overflow}\}$
- $\rightarrow = \{(i, i + 1) : 0 \leq i < 99\}$
- $\cup \{(99, \text{overflow})\}$
- $\cup \{(\text{overflow}, \text{overflow})\}$
- $s_0 = 0$
- Deterministic
Example: yes

```c
void yes() {
    while(true) print("y\n");
}
```

\[ S = \{ s_0 \} \]

\[ L = \text{the set of strings} \]

\[ s_0 \rightarrow "y\n" \rightarrow s_0 \]
Example: Diagonally moving robot

Example

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

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

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

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

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

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

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

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

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
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]
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
  - $(i, j) \rightarrow (5, j)$ [fill 5L jug]
  - $(i, j) \rightarrow (i, 3)$ [fill 3L 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

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

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

- \( S = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 0 \leq i \leq 5 \text{ and } 0 \leq j \leq 3\} \)
- \( s_0 = (0, 0) \)
- \( \rightarrow \) given by

- \((i, j) \rightarrow (5, j - 5 + i))\) 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.

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

\[ s_0 = (0, 0) \]

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

- \((i, j) \rightarrow (0, j)\) [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** (or **trace**) 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 \(i \geq 1\).
Runs and reachability

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

- a **run** (or **trace**) 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 \(i \geq 1\).

- A run is **maximal** if it cannot be extended; i.e., it is either infinite, or ends in a state from which there are no transitions.

\[\text{NB}\ s' \text{ is reachable from } s \text{ if there is a run from } s \text{ which contains } s'.\]
Runs and reachability

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

- a **run** (or **trace**) 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 \(i \geq 1\).

- A run is **maximal** if it cannot be extended; i.e., it is either infinite, or ends in a state from which there are no transitions.

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

**NB**

\(s'\) is reachable from \(s\) if there is a run from \(s\) which contains \(s'\).
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 \((0, 0)\)?
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 and Liveness

Transition systems can be used to study whether systems satisfy safety and liveness properties.

**Safety:** something bad will never happen.

**Liveness:** something good will happen.

Contrast this with reachability:

**Reachability:** something good can happen.
Example

Suppose our transition system models a nuclear power plant.

**Safety:** the reactor never reaches the *meltdown* state.

**Liveness:** the power plant will keep supplying power.
Safety and Liveness: Examples

Example

```java
void yes() {
    while(true){
        print("y\n");
    }
}
```

Safety: `yes()` never prints anything but "y\n".

Liveness: `yes()` will always print another "y\n".
Example

\[
y := 1; \\
z := x; \\
\text{while}(z \neq 0)\{ \\
y := y \times z; \\
z := z - 1; \\
\}\]

**Safety:** If the program ever terminates, then \( y = x! \)

**Liveness:** The program will terminate

(How is that a safety property?)
A property is a set of infinite runs. (Terminating runs can be made infinite by adding a self-loop to the final state.)

**Safety:** A safety property can be falsified by a finite prefix of a behaviour.

**Liveness:** A liveness property can always be satisfied eventually.
Properties Examples

Are they safety or liveness?

- *When I come home, there must be beer in the fridge*

- *I’ll be home later* (Liveness)

- *The program never allocates more than 100MB of memory* (Safety)

- *The program will allocate at least 100MB of memory* (Liveness)

- *No two processes are simultaneously in their critical section* (Safety)

- *If a process wishes to enter its critical section, it will eventually be allowed to do so* (Liveness)
Properties Examples

Are they safety or liveness?

- *When I come home, there must be beer in the fridge* – **Safety**
- *When I come home, I’ll drop on the couch and drink a beer*
Properties Examples

Are they safety or liveness?

- *When I come home, there must be beer in the fridge* – **Safety**
- *When I come home, I’ll drop on the couch and drink a beer* – **Liveness**
- *I’ll be home later* – **Liveness**
- *The program never allocates more than 100MB of memory*
Properties Examples

Are they safety or liveness?

- When I come home, there must be beer in the fridge – **Safety**
- When I come home, I’ll drop on the couch and drink a beer – **Liveness**
- I’ll be home later – **Liveness**
- The program never allocates more than 100MB of memory — **Safety**
- The program will allocate at least 100MB of memory
Properties Examples

Are they safety or liveness?

- *When I come home, there must be beer in the fridge* – **Safety**
- *When I come home, I’ll drop on the couch and drink a beer* – **Liveness**
- *I’ll be home later* – **Liveness**
- *The program never allocates more than 100MB of memory* — **Safety**
- *The program will allocate at least 100MB of memory* – **Liveness**
- *No two processes are simultaneously in their critical section*
Properties Examples

Are they safety or liveness?

- *When I come home, there must be beer in the fridge* – **Safety**
- *When I come home, I’ll drop on the couch and drink a beer* – **Liveness**
- *I’ll be home later* – **Liveness**
- *The program never allocates more than 100MB of memory* — **Safety**
- *The program will allocate at least 100MB of memory* – **Liveness**
- *No two processes are simultaneously in their critical section* — **Safety**
- *If a process wishes to enter its critical section, it will eventually be allowed to do so*
Properties Examples

Are they safety or liveness?

- *When I come home, there must be beer in the fridge* – **Safety**
- *When I come home, I’ll drop on the couch and drink a beer* – **Liveness**
- *I’ll be home later* – **Liveness**
- *The program never allocates more than 100MB of memory* — **Safety**
- *The program will allocate at least 100MB of memory* – **Liveness**
- *No two processes are simultaneously in their critical section* — **Safety**
- *If a process wishes to enter its critical section, it will eventually be allowed to do so* – **Liveness**
Summary

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

\[ \text{isBlue}((m, n)) := 2 | (m + n) \]
Safety example: Diagonally moving robot

Example

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

\( \text{isBlue}((m, n)) := 2 | (m + n) \)

if \( \text{isBlue}(s) \) and \( s \rightarrow s' \)
then \( \text{isBlue}(s') \)
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 \rightarrow s'
then isBlue(s')

isBlue((0, 0)) and \neg isBlue((0, 1))
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 \( s \).
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 sketch: Let $s'$ be a state reachable from $s$. We can show $\varphi(s')$ by induction on the length of the run from $s$ to $s'$.
Invariant example: Modified Die Hard problem

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 \((0, 0)\)?
Invariant example: Modified Die Hard problem

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 is a safety property. It doesn’t say whether the transition system will ever reach a final state.*
Partial correctness example: Fast exponentiation

Example

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

\begin{verbatim}
x := m;
y := n;
r := 1;
while y > 0 do
  if 2|y then
    y := y/2
  else
    y := (y - 1)/2;
r := r * x
fi;
x := x * x
od
\end{verbatim}
Partial correctness example: Fast exponentiation

Example

- States: Functions from \( \{m, n, x, y, r\} \) to \( \mathbb{N} \)
- Transitions:

Start state: \((m, n, 1)\)

Final states: \(\{x, 0, r\} : x, r \in \mathbb{N}\)

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

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?
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

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

Show \(\psi((x, y, r)) := (rx^y = m^x)\) 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:** \((m, n, 1)\)

Goal: Show partial correctness for \(\phi((x, y, r)) := (r = mn)\)

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: \(\{(x, 0, r) : x, r \in \mathbb{N}\}\)
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 \(\varphi((x, y, r)) := (r = m^n)\)
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...
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?
Total correctness = safety + liveness

A transition system \((S, \rightarrow)\) terminates from a state \(s \in S\) if all runs from \(s\) have finite length.

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

In a transition system \((S, \rightarrow)\), a **measure** is a function \(f : S \rightarrow \mathbb{N}\).

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

In a transition system \((S, \rightarrow)\), a measure is a function \(f : S \rightarrow \mathbb{N}\).

A measure is strictly decreasing if \(s \rightarrow s'\) implies \(f(s') < f(s)\).

**Theorem**

If \(f\) is a strictly decreasing measure, then the length of any run from \(s\) is at most \(f(s)\).
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

Measure:
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

**Measure:** \(f((x, y, r)) = y\)
Summary

- Motivation
- Definitions
- The invariant principle
- Partial correctness and termination
- Input and output
- Finite automata
We can model the system interacting with an external entity via inputs \((\Sigma)\) and outputs \((\Gamma)\) by using **labelled transitions**:

\[ \rightarrow \subseteq S \times L \times S \text{ where } L = \Sigma \times \Gamma \]

We’ll look at categories of input/output transition systems:

**Acceptors**: Accept/reject a sequence of inputs

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

We can model the system interacting with an external entity via inputs ($\Sigma$) and outputs ($\Gamma$) by using **labelled transitions**: $\rightarrow \subseteq S \times L \times S$ where $L = \Sigma \times \Gamma$

We’ll look at 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
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

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

\[ s_0 = (0, 0) \]

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

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

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

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

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
Example

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

\[ s_0 = (0, 0) \]

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

- \( (i, j) \xrightarrow{E_5} (0, j) \) [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:
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
  - \((i, j) \xrightarrow{E_5} (0, j)\) \hspace{1cm} [empty 5L jug]
  - \((i, j) \xrightarrow{E_3} (i, 0)\) \hspace{1cm} [empty 3L jug]
  - \((i, j) \xrightarrow{F_5} (5, j)\) \hspace{1cm} [fill 5L jug]
  - \((i, j) \xrightarrow{F_3} (i, 3)\) \hspace{1cm} [fill 3L jug]
  - \((i, j) \xrightarrow{E_{35}} (i + j, 0) \text{ if } i + j \leq 5\) \hspace{1cm} [empty 3L jug into 5L jug]
  - \((i, j) \xrightarrow{E_{53}} (0, i + j) \text{ if } i + j \leq 3\) \hspace{1cm} [empty 5L jug into 3L jug]
  - \((i, j) \xrightarrow{F_{35}} (5, j - 5 + i) \text{ if } i + j \geq 5\) \hspace{1cm} [fill 5L jug from 3L jug]
  - \((i, j) \xrightarrow{F_{35}} (i - 3 + j, 3) \text{ if } i + j \geq 3\) \hspace{1cm} [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.
Formal definitions

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
Finite state transition systems

State transition systems with a finite set of states are particularly useful in Computer Science.

**Acceptors:** Finite state automata

**Transducers:** Mealy machines