Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

Concrete model of computation

Dr. Giuseppe Maggiore

Hogeschool Rotterdam
Rotterdam, Netherlands

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

Lecture topics

- We discuss a formal way to define computation
- We discuss the fundamental elements of a concrete computer
- We bridge what we have seen in the previous lecture with concrete descriptions

Semantics

- Any language has **semantics**
- **Semantics** describe the *meaning* of sentences in the language
- Programming languages have **formal semantics**
- **Formal semantics** are expressed in a very logical, unambiguous format

HOGESCHOOL
ROTTERDAM

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

Consider this program from the previous lecture:

```
1  take 3 steps forward
2  sit on the chair
3  turn left
4  slide 3 steps forward
```

What do you implicitly assume by performing each of the
instructions? **Try to guess and discuss!**

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

Semantics of `stdNt`

- We start with a *current instruction* and a *student state*:
  - The current instruction (often called *instruction pointer* (IP) or *program counter* (PC)) is just the index of the current instruction;
  - the student state (usually just called *state*, or S, or σ) is whatever relevant attributes we track about the student (for example, his position and orientation in the room and whether or not he is sitting).
- Each instruction changes the PC and the S.

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
1	Standing	Forward	(0,0)

```
1  take 3 steps forward
2  sit on the chair
3  turn left
4  slide 3 steps forward
```

what changes while running the current instruction? **Try to guess and discuss!**

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
2	Standing	Forward	(0,3)

```
1  take 3 steps forward
2  sit on the chair
3  turn left
4  slide 3 steps forward
```

what changes while running the current instruction? **Try to guess and discuss!**

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
3	Sitting	Forward	(0,3)

```
1  take 3 steps forward
2  sit on the chair
3  turn left
4  slide 3 steps forward
```

what changes while running the current instruction? **Try to guess and discuss!**

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
4	Sitting	Left	(0,3)

```
1  take 3 steps forward
2  sit on the chair
3  turn left
4  slide 3 steps forward
```

what changes while running the current instruction? **Try to
guess and discuss!**

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
END	Sitting	Left	(-3,3)

```
1  take 3 steps forward
2  sit on the chair
3  turn left
4  slide 3 steps forward
```

what do we do now? **Try to guess and discuss!**

A slight formalization

- We say that an instruction I is a *function* that, given a pair of PC and S, returns a new pair of PC and S

A slight formalization

- We say that an instruction I is a *function* that, given a pair of PC and S, returns a new pair of PC and S
- Do not panic now, math..y symbols incoming!

A slight formalization

- We say that an instruction I is a *function* that, given a pair of PC and S, returns a new pair of PC and S
- Do not panic now, math..y symbols incoming!
- $(PC, S) \overset{Instr}{\to} (PC', S')$

A slight formalization

- Consider instruction `sit on the chair` (we will shorten it to `sit`)
- *How do we change the current instruction?*
- *How do we change the position of the resulting state depending on the orientation of the input state?*

A slight formalization

- Consider instruction `sit on the chair` (we will shorten it to `sit`)

    - $(PC, S) \stackrel{sit}{\to} (PC + 1, S[Pose \mapsto Sitting])$

- We increment the current instruction index by one

- We change the pose of the resulting state independent on the input state

    - $S[Pose \mapsto Sitting]$ is read as *"S, where pose is sitting"*

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `stand up` (we will shorten it to `stand`)
- *How do we change the current instruction?*
- *How do we change the position of the resulting state depending on the orientation of the input state?*

A slight formalization

- Consider instruction `stand up` (we will shorten it to `stand`)

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `stand up` (we will shorten it to `stand`)
  - $(PC, S) \overset{stand}{\rightarrow} (PC + 1, S[Pose \mapsto Standing])$
- We increment the current instruction index by one
- We change the pose of the resulting state independent on the input state

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `take 3 steps forward` (we will shorten it to `fwd 3`)
- *How do we determine the next instruction index?*
- *How do we change the position of the resulting state?*
  - Are there dependencies from the input state?

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
104	Standing	Left	(10,20)

```
103  ...
104  take 3 steps forward
105  ...
```

PC	S.Pose	S.Orientation	S.Position
105	Standing	Left	(7,20)

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
104	Standing	Right	(10,20)

```
103   ...
104   take 3 steps forward
105   ...
```

PC	S.Pose	S.Orientation	S.Position
105	Standing	Right	(13,20)

A slight formalization

- Consider instruction `take 3 steps forward` (we will shorten it to `fwd 3`)

$$
\begin{cases}
(PC, S) \xrightarrow{fwd3} (PC + 1, S[Position \mapsto S.Position + (0, 3)]) \\
when\ S.Orientation = Forward \\
(PC, S) \xrightarrow{fwd3} (PC + 1, S[Position \mapsto S.Position - (0, 3)]) \\
when\ S.Orientation = Backward \\
(PC, S) \xrightarrow{fwd3} (PC + 1, S[Position \mapsto S.Position + (3, 0)]) \\
whenS.Orientation = Right \\
(PC, S) \xrightarrow{fwd3} (PC + 1, S[Position \mapsto S.Position - (3, 0)]) \\
whenS.Orientation = Left
\end{cases}
$$

- We always increment the instruction by one
- We change the position of the resulting state depending on the orientation of the input state

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `if A then B else C`
- *How do we determine the next instruction index?*
- *How do we change the state?*

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
24	Standing	Right	(10,20)

```
23 | ...
24 | if A is ``black'' then
25 |   turn left by 90 * B degrees
26 | otherwise
27 |   turn left by 90 * C degrees
28 | ...
```

PC	S.Pose	S.Orientation	S.Position
25[1]	Standing	Right	(10,20)

[1]Assuming student's shirt is black

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
24	Standing	Right	(10,20)

```
23   ...
24   if A is ''black'' then
25     turn left by 90 * B degrees
26   otherwise
27     turn left by 90 * C degrees
28   ...
```

PC	S.Pose	S.Orientation	S.Position
27[2]	Standing	Right	(10,20)

[2]Assuming student's shirt is not black

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `if A then B else C` (shortened by as if_{ABC})

A slight formalization

- Consider instruction `if A then B else C` (shortened by as if_{ABC})
- We jump to the first instruction of the `B` block if the condition evaluates to `TRUE`

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `if A then B else C` (shortened by as if_{ABC})
- We jump to the first instruction of the `B` block if the condition evaluates to `TRUE`
- We jump to the first instruction of the `C` block if the condition evaluates to `FALSE`

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `if A then B else C` (shortened by as if_{ABC})
- We jump to the first instruction of the `B` block if the condition evaluates to `TRUE`
- We jump to the first instruction of the `C` block if the condition evaluates to `FALSE`
- We leave the state unchanged

HOGESCHOOL
ROTTERDAM

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

Introduction

A slight formalization

- Consider instruction `if A then B else C` (shortened by as if_{ABC})
- We jump to the first instruction of the `B` block if the condition evaluates to `TRUE`
- We jump to the first instruction of the `C` block if the condition evaluates to `FALSE`
- We leave the state unchanged

$$\begin{cases} (PC, S) \stackrel{if_{ABC}}{\rightarrow} (loc(B), S) & when & (PC, S) \stackrel{A}{\rightarrow} \texttt{TRUE} \\ (PC, S) \stackrel{if_{ABC}}{\rightarrow} (loc(C), S) & when & (PC, S) \stackrel{A}{\rightarrow} \texttt{FALSE} \end{cases}$$

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `while A do B`
- *How do we determine the next instruction index?*
- *How do we change the state?*

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
24	Standing	Right	(10,20)

```
23  ...
24  while A is ''sunny'' do
25    enjoy the day for another hour
26  go back to work
27  ...
```

PC	S.Pose	S.Orientation	S.Position
25^3	Standing	Right	(10,20)

[3]As long as it is sunny

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

PC	S.Pose	S.Orientation	S.Position
24	Standing	Right	(10,20)

```
23  ...
24  while A is ''sunny'' do
25    enjoy the day for another hour
26  go back to work
27  ...
```

PC	S.Pose	S.Orientation	S.Position
26[4]	Standing	Right	(10,20)

[4] When it stops being sunny

Concrete
model of
computation

Dr. Giuseppe
Maggiore

A slight formalization

- Consider instruction `while A do B` (shortened by as $while_{AB}$)

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `while A do B` (shortened by as $while_{AB}$)
- We jump to the first instruction of the B block if the condition evaluates to `TRUE`

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

A slight formalization

- Consider instruction `while A do B` (shortened by as $while_{AB}$)
- We jump to the first instruction of the B block if the condition evaluates to `TRUE`
- We jump to after the last instruction of the B block if the condition evaluates to `FALSE`

A slight formalization

- Consider instruction `while A do B` (shortened by as $while_{AB}$)
- We jump to the first instruction of the B block if the condition evaluates to `TRUE`
- We jump to after the last instruction of the B block if the condition evaluates to `FALSE`
- We leave the state unchanged

Concrete
model of
computation

Dr. Giuseppe
Maggiore

Introduction

Introduction

A slight formalization

- Consider instruction `while A do B` (shortened by as $while_{AB}$)
- We jump to the first instruction of the `B` block if the condition evaluates to `TRUE`
- We jump to after the last instruction of the `B` block if the condition evaluates to `FALSE`
- We leave the state unchanged

$$\begin{cases} (PC, S) \stackrel{while_{AB}}{\rightarrow} (loc(B), S) \ when \ (PC, S) \stackrel{A}{\rightarrow} \texttt{TRUE} \\ (PC, S) \stackrel{while_{AB}}{\rightarrow} (lastloc(B) + 1, S) \ when \ (PC, S) \stackrel{A}{\rightarrow} \texttt{FALSE} \end{cases}$$

Concrete model of computation

Dr. Giuseppe Maggiore

Introduction

The best of luck, and thanks for the attention!