Programming Languages and Compilers (CS 421)

Talia Ringer (they/them)

4218 SC, UIUC

https://courses.grainger.illinois.edu/cs421/fa2023/

Based heavily on slides by Elsa Gunter, which were based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

Midterm Study Guide

Objectives for Today

Three Main Topics of the Course

I

New Programming Paradigm

II

Language Translation

III

Language Semantics

Objectives for Today

Three Main Topics of the Course

I

New Programming Paradigm

II

Language Translation

III

Language Semantics

Objectives for Today

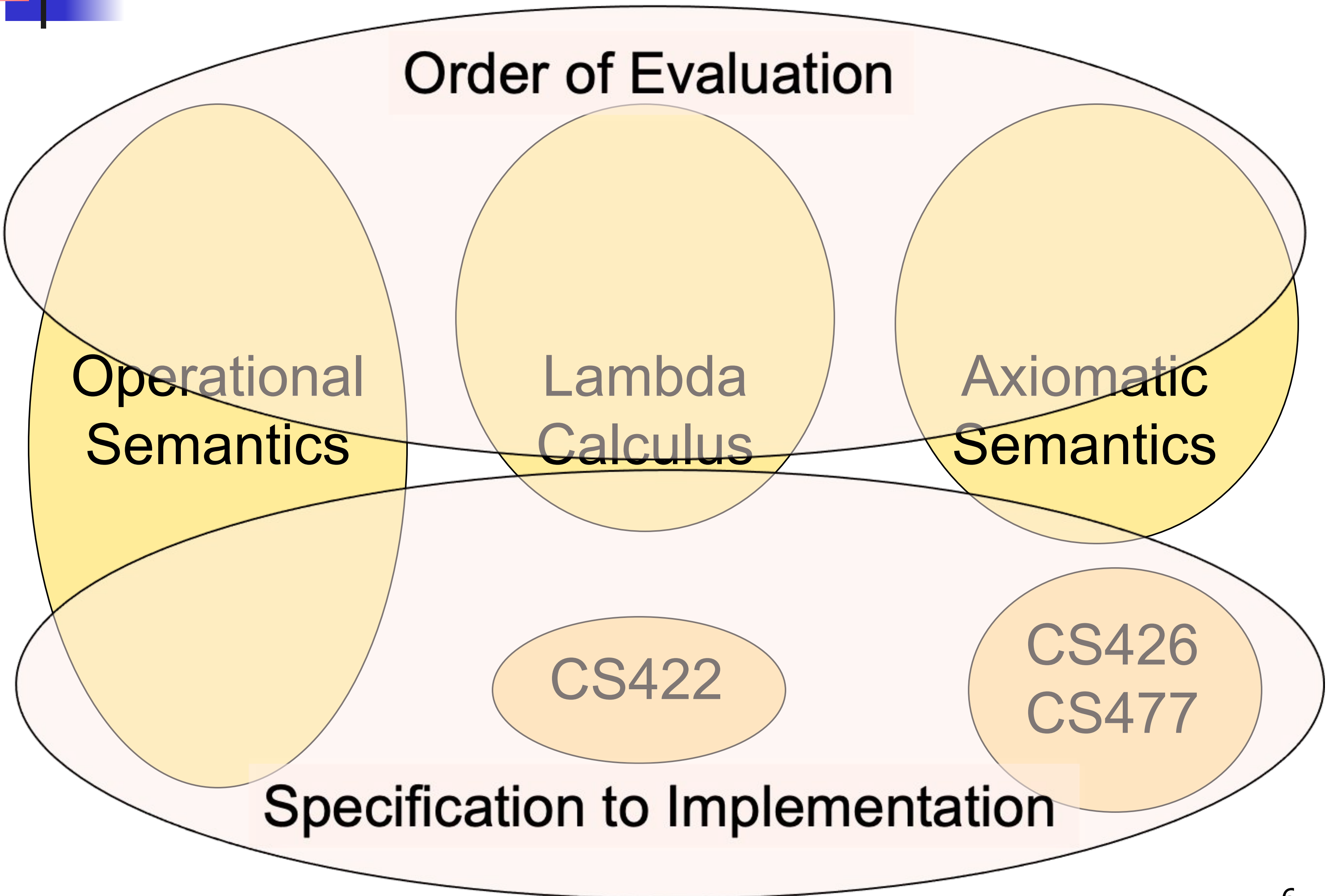
III : Language Semantics

Operational Semantics

Lambda Calculus

Axiomatic Semantics

Objectives for Today



Questions before we start?

Semantics

Semantics

- Expresses the **meaning** of syntax
- **Static** semantics:
  - Meaning based only on the form of the expression **without executing** it
  - Usually restricted to **type checking** / **type inference**
- **Dynamic** semantics:
  - Describes meaning of **executing** a program
  - Kinds: **operational**, **axiomatic**, **denotational**

Semantics

- Expresses the **meaning** of syntax
- **Static** semantics:
  - Meaning based only on the form of the expression **without executing** it
  - Usually restricted to **type checking** / **type inference**
- **Dynamic** semantics:
  - Describes meaning of **executing** a program
  - Kinds: **operational**, **axiomatic**, **denotational**

Semantics

- Expresses the **meaning** of syntax
- **Static** semantics:
  - Meaning based only on the form of the expression **without executing** it
  - Usually restricted to **type checking** / **type inference**
- **Dynamic** semantics:
  - Describes meaning of **executing** a program
  - Kinds: **operational**, **axiomatic**, **denotational**

Semantics

- Expresses the **meaning** of syntax
- **Static** semantics:
  - Meaning based only on the form of the expression **without executing** it
  - Usually restricted to **type checking** / **type inference**
- **Dynamic** semantics:
  - Describes meaning of **executing** a program
  - Kinds: **operational**, **axiomatic**, **denotational**

Dynamic Semantics

- Why so many **kinds** of dynamic semantics?
  - **Different languages** better suited to different kinds of semantics
  - Different kinds serve **different purposes**
  - Common to have **multiple** kinds and show how they **relate** to each other

- **Dynamic** semantics:
  - Describes meaning of **executing** a program
  - Kinds: **operational**, **axiomatic**, **denotational**

Operational Semantics

- **What it is:**
  - Describe how to execute (implement) programs of language on a virtual machine, by describing **how to execute each program statement** (i.e., following the **structure** of the program)
  - Meaning of program is how its execution **changes the state** of the machine
- **Tradeoffs:**
  - Easy to **implement**
  - Hard to **reason about abstractly** (without thinking about implementation details)

Operational Semantics

- **What it is:**
  - Describe how to execute (implement) programs of language on a virtual machine, by describing **how to execute each program statement** (i.e., following the **structure** of the program)
  - Meaning of program is how its execution **changes the state** of the machine
- **Tradeoffs:**
  - Easy to **implement**
  - Hard to **reason about abstractly** (without thinking about implementation details)

Axiomatic Semantics

- **What it is**:
  - Also called a **Program Logic**
    - Commonly **Floyd-Hoare logic**
    - These days, also **separation logic**
  - Logical system built from ***axioms*** and ***inference rules***
  - Often written as **pre-conditions** and **post-conditions** on programs
- **Tradeoffs:**
  - Mainly suited to **imperative languages**
  - Good for **external** reasoning

Axiomatic Semantics

- **What it is**:
  - Also called a **Program Logic**
    - Commonly **Floyd-Hoare logic**
    - These days, also **separation logic**
  - Logical system built from ***axioms*** and ***inference rules***
  - Often written as **pre-conditions** and **post-conditions** on programs
- **Tradeoffs:**
  - Mainly suited to **imperative languages**
  - Good for **external** reasoning

Axiomatic Semantics

- **What it is**:
  - Also called a **Program Logic**
    - Commonly **Floyd-Hoare logic**
    - These days, also **separation logic**
  - Logical system built from ***axioms*** and ***inference rules***
  - Often written as **pre-conditions** and **post-conditions** on programs
- **Tradeoffs:**
  - Mainly suited to **imperative languages**
  - Good for **external** reasoning

Axiomatic Semantics

- Used to formally prove a **post-condition** (property) of the **state** (the values of the program variables) after the execution of program, assuming a **pre-condition** (another property) holds before execution
- Written :

    {Precondition} Program {Postcondition}
- Source of idea of **loop invariant**

Axiomatic Semantics

- Used to formally prove a **post-condition** (property) of the **state** (the values of the program variables) after the execution of program, assuming a **pre-condition** (another property) holds before execution
- Written :

  {Precondition} Program {Postcondition}
- Source of idea of **loop invariant**

Axiomatic Semantics

- Used to formally prove a **post-condition** (property) of the **state** (the values of the program variables) after the execution of program, assuming a **pre-condition** (another property) holds before execution
- Written :

  {Precondition} Program {Postcondition}
- Source of idea of **loop invariant**

Denotational Semantics

- **What it is:**
  - Construct function $\mathcal{M}$ assigning **mathematical meaning** to each program construct
    - via category theory, algebra, probability theory, topology, lambda calculus, …
  - Meaning function is **compositional**: meaning of construct built from meaning of parts
- **Tradeoffs:**
  - Useful for **proving** properties of programs
  - Doesn't help much with **implementation**

Denotational Semantics

- **What it is:**
  - Construct function $\mathcal{M}$ assigning **mathematical meaning** to each program construct
    - via category theory, algebra, probability theory, topology, lambda calculus, …
  - Meaning function is **compositional**: meaning of construct built from meaning of parts
- **Tradeoffs:**
  - Useful for **proving** properties of programs
  - Doesn’t help much with **implementation**

Operational Semantics

- **What it is:**
  - Describe how to execute (implement) programs of language on a virtual machine, by describing **how to execute each program statement** (i.e., following the **structure** of the program)
  - Meaning of program is how its execution **changes the state** of the machine
- **Tradeoffs:**
  - Easy to **implement**
  - Hard to **reason about abstractly** (without thinking about implementation details)

Operational Semantics

- Can be **small step** or **big step**
  - **Small step:** define meaning of one step of execution of a program statement at a time
  - **Big step:** define meaning in terms of value of execution of whole program statement
- **Common to have both** and **relate** them

Operational Semantics

- Can be **small step** or **big step**
  - **Small step:** define meaning of one step of execution of a program statement at a time
  - **Big step:** define meaning in terms of value of execution of whole program statement
- **Common to have both** and **relate** them

Natural (Big Step) Semantics

Natural Semantics

- Also known as **Structural Operational Semantics** or **Big Step Semantics**
- Provide **value** for a program by **rules** and **derivations**, similar to type derivations
- Rule conclusions look like:

$$(C, m) \Downarrow m'$$

or

$$(E, m) \Downarrow v$$

Natural Semantics

- Also known as **Structural Operational Semantics** or **Big Step Semantics**
- Provide **value** for a program by **rules** and **derivations**, similar to type derivations
- Rule conclusions look like:

$$(C, m) \Downarrow m'$$

or

$$(E, m) \Downarrow v$$

Simple Imperative Language Syntax

$I \in$ Identifiers

$N \in$ Numerals

B ::= true | false | **B** & **B** | **B** or **B** |
  not **B** | **E** < **E** | **E** = **E**

E ::= N | I | **E** + **E** | **E** * **E** | **E** - **E** | - **E** | (**E**)

C ::= skip | **C**; **C** | I := **E** |
  if **B** then **C** else **C** fi | while **B** do **C** od

Simple Imperative Language Syntax

$I \in$ Identifiers

$N \in$ Numerals

B ::= true | false | **B** & **B** | **B** or **B** |
not **B** | **E** < **E** | **E** = **E**

E ::= N | I | **E** + **E** | **E** * **E** | **E** - **E** | - **E** | (**E**)

C ::= skip | **C**; **C** | I := **E** |
if **B** then **C** else **C** fi | while **B** do **C** od

Simple Imperative Language Syntax

I ∈ Identifiers

N ∈ Numerals

B ::= true | false | **B** & **B** | **B** or **B** |
not **B** | **E** < **E** | **E** = **E**

E ::= N | I | **E** + **E** | **E** * **E** | **E** - **E** | - **E** | (**E**)

C ::= skip | **C**; **C** | I := **E** |
if **B** then **C** else **C** fi | while **B** do **C** od

Simple Imperative Language Syntax

I ∈ Identifiers

N ∈ Numerals

B ::= true | false | **B** & **B** | **B** or **B** |
not **B** | **E** < **E** | **E** = **E**

E ::= N | I | **E** + **E** | **E** * **E** | **E** - **E** | - **E** | (**E**)

C ::= skip | **C**; **C** | I := **E** |
if **B** then **C** else **C** fi | while **B** do **C** od

Simple Imperative Language Semantics

$$(E, m) \Downarrow v$$

Look up identifiers

$$\frac{}{(I, m) \Downarrow m(I)}\ \textbf{Id}$$

$$\frac{}{(N, m) \Downarrow N}\ \textbf{Num}$$

$$\frac{}{(\text{true}, m) \Downarrow \text{true}}\ \textbf{True}$$

$$\frac{}{(\text{false}, m) \Downarrow \text{false}}\ \textbf{False}$$

Natural Semantics

Simple Imperative Language Semantics

$(E, m) \Downarrow v$

$$\frac{}{(I, m) \Downarrow m(I)}\ \textbf{Id}$$

Numerals are **literals**

$$\frac{}{(N, m) \Downarrow N}\ \textbf{Num}$$

$$\frac{}{(\text{true}, m) \Downarrow \text{true}}\ \textbf{True} \qquad \frac{}{(\text{false}, m) \Downarrow \text{false}}\ \textbf{False}$$

Natural Semantics

Simple Imperative Language Semantics

$$\boxed{(E, m) \Downarrow v}$$

$$\frac{}{(I, m) \Downarrow m(I)}\ \textbf{Id}$$

$$\frac{}{(N, m) \Downarrow N}\ \textbf{Num}$$

$$\boxed{(B, m) \Downarrow v}$$

$$\frac{}{(\text{true}, m) \Downarrow \text{true}}\ \textbf{True} \qquad \frac{}{(\text{false}, m) \Downarrow \text{false}}\ \textbf{False}$$

Boolean atoms
are **literals** too

Questions so far?

Simple Imperative Language Semantics

$(B, m) \Downarrow v$

$$\frac{(B, m) \Downarrow \text{false}}{(B \;\&\; B', m) \Downarrow \text{false}}\;\textbf{And-F} \qquad \frac{(B, m) \Downarrow \text{true} \quad (B', m) \Downarrow b}{(B \;\&\; B', m) \Downarrow b}\;\textbf{And-T}$$

$$\frac{(B, m) \Downarrow \text{true}}{(B \text{ or } B', m) \Downarrow \text{true}}\;\textbf{Or-T} \qquad \frac{(B, m) \Downarrow \text{false} \quad (B', m) \Downarrow b}{(B \text{ or } B', m) \Downarrow b}\;\textbf{Or-F}$$

$$\frac{(B, m) \Downarrow \text{true}}{(\text{not } B, m) \Downarrow \text{false}}\;\textbf{Not-T} \qquad \frac{(B, m) \Downarrow \text{false}}{(\text{not } B, m) \Downarrow \text{true}}\;\textbf{Not-F}$$

Boolean combinators have the **standard** meaning

Natural Semantics

Simple Imperative Language Semantics

$$(B, m) \Downarrow v$$

$$\frac{(B, m) \Downarrow \text{false}}{(B \mathbin{\&} B', m) \Downarrow \text{false}}\ \textbf{And-F} \qquad \frac{(B, m) \Downarrow \text{true} \quad (B', m) \Downarrow b}{(B \mathbin{\&} B', m) \Downarrow b}\ \textbf{And-T}$$

$$\frac{(B, m) \Downarrow \text{true}}{(B \text{ or } B', m) \Downarrow \text{true}}\ \textbf{Or-T} \qquad \frac{(B, m) \Downarrow \text{false} \quad (B', m) \Downarrow b}{(B \text{ or } B', m) \Downarrow b}\ \textbf{Or-F}$$

$$\frac{(B, m) \Downarrow \text{true}}{(\text{not } B, m) \Downarrow \text{false}}\ \textbf{Not-T} \qquad \frac{(B, m) \Downarrow \text{false}}{(\text{not } B, m) \Downarrow \text{true}}\ \textbf{Not-F}$$

Boolean combinators have the **standard** meaning

Natural Semantics

Simple Imperative Language Semantics

$$(B, m) \Downarrow v$$

$$\frac{(B, m) \Downarrow \text{false}}{(B \;\&\; B', m) \Downarrow \text{false}}\ \textbf{And-F} \qquad \frac{(B, m) \Downarrow \text{true} \quad (B', m) \Downarrow b}{(B \;\&\; B', m) \Downarrow b}\ \textbf{And-T}$$

$$\frac{(B, m) \Downarrow \text{true}}{(B \text{ or } B', m) \Downarrow \text{true}}\ \textbf{Or-T} \qquad \frac{(B, m) \Downarrow \text{false} \quad (B', m) \Downarrow b}{(B \text{ or } B', m) \Downarrow b}\ \textbf{Or-F}$$

$$\frac{(B, m) \Downarrow \text{true}}{(\text{not } B, m) \Downarrow \text{false}}\ \textbf{Not-T} \qquad \frac{(B, m) \Downarrow \text{false}}{(\text{not } B, m) \Downarrow \text{true}}\ \textbf{Not-F}$$

Boolean combinators have the **standard** meaning

Natural Semantics

Simple Imperative Language Semantics

$(E, m) \Downarrow v$

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \sim V = b}{(E \sim E', m) \Downarrow b} \text{Rel}$$

- By **U ~ V = b**, we mean: does (the meaning of) the relation ~ hold on the meaning of U and V?
- May be specified by a mathematical expression/equation or rules matching U and V

Relations like <, >, and = are defined in terms of their **primitive** meanings

Simple Imperative Language Semantics

$(E, m) \Downarrow v$

$$\frac{(E, m) \Downarrow U \qquad (E', m) \Downarrow V \qquad \mathbf{U \sim V = b}}{(E \sim E', m) \Downarrow b}\ \textbf{Rel}$$

- By **U ~ V = b**, we mean: does (the meaning of) the relation **~** hold on the meaning of **U** and **V**?
- May be specified by a mathematical expression/equation or rules matching **U** and **V**

Relations like <, >, and = are defined in terms of their **primitive** meanings

Simple Imperative Language Semantics

$(E, m) \Downarrow v$

$$\frac{(\mathbf{E}, \mathbf{m}) \Downarrow \mathbf{U} \qquad (E', m) \Downarrow V \qquad \mathbf{U} \sim \mathbf{V} = \mathbf{b}}{(\mathbf{E} \sim E', m) \Downarrow b} \;\textbf{Rel}$$

- By **U ~ V = b**, we mean: does (the meaning of) the relation **~** hold on the meaning of **U** and **V**?
- May be specified by a mathematical expression/equation or rules matching **U** and **V**

Relations like <, >, and = are defined in terms of their **primitive** meanings

Simple Imperative Language Semantics

$(E, m) \Downarrow v$

$$\frac{(\mathbf{E}, \mathbf{m}) \Downarrow \mathbf{U} \quad (\mathbf{E'}, \mathbf{m}) \Downarrow \mathbf{V} \quad \mathbf{U} \sim \mathbf{V} = \mathbf{b}}{(\mathbf{E} \sim \mathbf{E'}, m) \Downarrow b} \ \mathbf{Rel}$$

- By **U ~ V = b**, we mean: does (the meaning of) the relation **~** hold on the meaning of **U** and **V**?
- May be specified by a mathematical expression/equation or rules matching **U** and **V**

Relations like <, >, and = are defined in terms of their **primitive** meanings

Simple Imperative Language Semantics

$(E, m) \Downarrow v$

$$\frac{(E, m) \Downarrow U \qquad (E', m) \Downarrow V \qquad \mathbf{U \sim V = b}}{(E \sim E', m) \Downarrow b}\ \textbf{Rel}$$

- By **U ~ V = b**, we mean: does (the meaning of) the relation **~** hold on the meaning of **U** and **V**?
- May be specified by a mathematical expression/equation or rules matching **U** and **V**

Relations like <, >, and = are defined in terms of their **primitive** meanings

Simple Imperative Language Semantics

$$(E, m) \Downarrow v$$

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \text{ op } V = N}{(E \text{ op } E', m) \Downarrow N}\ \textbf{Arith}$$

where **N** is the specified value for **U op V**

Arithmetic expressions are defined **similarly**

Simple Imperative Language Semantics

$(E, m) \Downarrow v$

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad \mathbf{U\ op\ V = N}}{(E\ op\ E', m) \Downarrow N}\ \mathbf{Arith}$$

where **N** is the specified value for **U op V**

Arithmetic expressions are defined **similarly**

Questions so far?

Simple Imperative Language Semantics

$(C, m) \Downarrow \mathbf{m'}$

$$\frac{}{(\text{skip}, m) \Downarrow m}\ \textbf{Skip}$$

Commands evaluate to **maps of variables** (environments or stacks) rather than to **values**

$$\frac{(E, m) \Downarrow v}{(I := E, m) \Downarrow m[I <\!\!- v]}\ \textbf{Assign}$$

$$\frac{(C, m) \Downarrow m' \quad (C', m') \Downarrow m''}{(C; C', m) \Downarrow m''}\ \textbf{Seq}$$

Natural Semantics

Simple Imperative Language Semantics

$(C, m) \Downarrow m'$

$$\frac{}{(\text{skip}, m) \Downarrow m}\ \textbf{Skip}$$

Skip doesn't change the state

$$\frac{(E, m) \Downarrow v}{(I := E, m) \Downarrow m[I<-v]}\ \textbf{Assign}$$

$$\frac{(C, m) \Downarrow m' \quad (C', m') \Downarrow m''}{(C; C', m) \Downarrow m''}\ \textbf{Seq}$$

Natural Semantics

Simple Imperative Language Semantics

$$(C, m) \Downarrow m'$$

$$\frac{}{(\mathrm{skip}, m) \Downarrow m}\ \textbf{Skip}$$

Assign updates the state with a **new mapping** of **identifier I** to **value v**

$$\frac{(E, m) \Downarrow v}{(I := E, m) \Downarrow m[I <- v]}\ \textbf{Assign}$$

$$\frac{(C, m) \Downarrow m' \quad (C', m') \Downarrow m''}{(C; C', m) \Downarrow m''}\ \textbf{Seq}$$

Natural Semantics

Simple Imperative Language Semantics

$(C, m) \Downarrow m'$

$$\frac{}{(\text{skip}, m) \Downarrow m}\ \textbf{Skip}$$

Assign updates the state with a **new mapping** of **identifier I** to **value v**

$$\frac{(\mathbf{E}, m) \Downarrow \mathbf{v}}{(I := \mathbf{E}, m) \Downarrow m[I \texttt{<-} \mathbf{v}]}\ \textbf{Assign}$$

$$\frac{(C, m) \Downarrow m' \quad (C', m') \Downarrow m''}{(C; C', m) \Downarrow m''}\ \textbf{Seq}$$

Natural Semantics

Simple Imperative Language Semantics

$$\boxed{(C, m) \Downarrow m'}$$

$$\frac{}{(\text{skip}, m) \Downarrow m}\ \textbf{Skip}$$

$$\frac{(E, m) \Downarrow v}{(I := E, m) \Downarrow m[I <\!\!- v]}\ \textbf{Assign}$$

Sequencing has the **usual meaning**

$$\frac{(C, m) \Downarrow m' \qquad (C', m') \Downarrow m''}{(C; C', m) \Downarrow m''}\ \textbf{Seq}$$

Natural Semantics

Simple Imperative Language Semantics

$\boxed{(C, m) \Downarrow m'}$

$$\frac{}{(\text{skip}, m) \Downarrow m}\ \textbf{Skip}$$

$$\frac{(E, m) \Downarrow v}{(I := E, m) \Downarrow m[I<-v]}\ \textbf{Assign}$$

Sequencing has the **usual meaning**

$$\frac{(\mathbf{C}, \mathbf{m}) \Downarrow \mathbf{m'} \quad (C', m') \Downarrow m''}{(\mathbf{C};\ C', \mathbf{m}) \Downarrow m''}\ \textbf{Seq}$$

Simple Imperative Language Semantics

$$(C, m) \Downarrow m'$$

$$\frac{}{(\text{skip}, m) \Downarrow m}\ \textbf{Skip}$$

$$\frac{(E, m) \Downarrow v}{(I := E, m) \Downarrow m[I{<}{-}v]}\ \textbf{Assign}$$

Sequencing has the **usual meaning**

$$\frac{(C, m) \Downarrow \mathbf{m'} \quad (\mathbf{C'}, \mathbf{m'}) \Downarrow \mathbf{m''}}{(C; \mathbf{C'}, m) \Downarrow m''}\ \textbf{Seq}$$

Natural Semantics

Simple Imperative Language Semantics

$$(C, m) \Downarrow m'$$

$$\frac{}{(\text{skip}, m) \Downarrow m}\ \textbf{Skip}$$

$$\frac{(E, m) \Downarrow v}{(I := E, m) \Downarrow m[I <\!\!- v]}\ \textbf{Assign}$$

Sequencing has the **usual meaning**

$$\frac{(C, m) \Downarrow m' \quad (C', m') \Downarrow \mathbf{m''}}{\mathbf{(C; C', m) \Downarrow m''}}\ \textbf{Seq}$$

Natural Semantics

Simple Imperative Language Semantics

$(C, m) \Downarrow m'$

If then else is split into two cases, one for **true** and one for **false**

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'} \ \textbf{If-T}$$

$$\frac{(B, m) \Downarrow \text{false} \quad (C', m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'} \ \textbf{If-F}$$

Simple Imperative Language Semantics

$(C, m) \Downarrow m'$

If then else is split into two cases, one for **true** and one for **false**

$$\frac{(B, m) \Downarrow \mathbf{true} \quad (\mathbf{C}, m) \Downarrow m'}{(\text{if } B \text{ then } \mathbf{C} \text{ else } C' \text{ fi}, m) \Downarrow m'}\ \textbf{If-T}$$

$$\frac{(B, m) \Downarrow \text{false} \quad (C', m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}\ \textbf{If-F}$$

Natural Semantics

Simple Imperative Language Semantics

$(C, m) \Downarrow m'$

If then else is split into two cases, one for **true** and one for **false**

$$\frac{(B, m) \Downarrow \textbf{true} \quad (\mathbf{C}, m) \Downarrow m'}{(\text{if } B \text{ then } \mathbf{C} \text{ else } C' \text{ fi}, m) \Downarrow m'}\ \textbf{If-T}$$

$$\frac{(B, m) \Downarrow \textbf{false} \quad (\mathbf{C'}, m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } \mathbf{C'} \text{ fi}, m) \Downarrow m'}\ \textbf{If-F}$$

Simple Imperative Language Semantics

$\boxed{(C, m) \Downarrow m'}$

$$\frac{(B, m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m}\ \textbf{While-F}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m' \quad (\text{while } B \text{ do } C \text{ od}, m') \Downarrow m''}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m''}\ \textbf{While-T}$$

While is likewise split into two cases, one for **true** and one for **false**

Natural Semantics

Simple Imperative Language Semantics

$(C, m) \Downarrow m'$

$$\frac{(B, m) \Downarrow \textbf{false}}{(\text{while } \textbf{B} \text{ do } C \text{ od}, m) \Downarrow m} \textbf{While-F}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m' \quad (\text{while } B \text{ do } C \text{ od}, m') \Downarrow m''}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m''} \textbf{While-T}$$

While is likewise split into two cases, one for **true** and one for **false**

Natural Semantics

Simple Imperative Language Semantics

$(C, m) \Downarrow m'$

$$\frac{(B, m) \Downarrow \mathbf{false}}{(\text{while } \mathbf{B} \text{ do } C \text{ od}, \mathbf{m}) \Downarrow \mathbf{m}} \ \textbf{While-F}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m' \quad (\text{while } B \text{ do } C \text{ od}, m') \Downarrow m''}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m''} \ \textbf{While-T}$$

While is likewise split into two cases, one for **true** and one for **false**

Natural Semantics

Simple Imperative Language Semantics

$\boxed{(C, m) \Downarrow m'}$

$$\frac{(B, m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m} \;\textbf{While-F}$$

$$\frac{\begin{array}{c}\mathbf{(B, m) \Downarrow true} \\ (C, m) \Downarrow m' \\ (\text{while } B \text{ do } C \text{ od}, m') \Downarrow m''\end{array}}{(\text{while } \mathbf{B} \text{ do } C \text{ od}, \mathbf{m}) \Downarrow m''} \;\textbf{While-T}$$

While is likewise split into two cases, one for **true** and one for **false**

Natural Semantics

Simple Imperative Language Semantics

$$(C, m) \Downarrow m'$$

$$\frac{(B, m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m}\ \textbf{While-F}$$

$$\frac{(B, m) \Downarrow \text{true} \quad \boldsymbol{(C, m) \Downarrow m'} \quad (\text{while } B \text{ do } C \text{ od}, m') \Downarrow m''}{(\text{while } B \text{ do } \mathbf{C} \text{ od}, \mathbf{m}) \Downarrow m''}\ \textbf{While-T}$$

While is likewise split into two cases, one for **true** and one for **false**

Natural Semantics

Simple Imperative Language Semantics

$\boxed{(C, m) \Downarrow m'}$

$$\frac{(B, m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m}\ \textbf{While-F}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow \mathbf{m'} \quad (\textbf{while B do C od}, \mathbf{m'}) \Downarrow m''}{(\textbf{while B do C od}, m) \Downarrow m''}\ \textbf{While-T}$$

While is likewise split into two cases, one for **true** and one for **false**

Natural Semantics

Simple Imperative Language Semantics

$$(C, m) \Downarrow m'$$

$$\frac{(B, m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m}\ \textbf{While-F}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m' \quad (\text{while } B \text{ do } C \text{ od}, m') \Downarrow \mathbf{m''}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow \mathbf{m''}}\ \textbf{While-T}$$

While is likewise split into two cases, one for **true** and one for **false**

Natural Semantics

Questions so far?

Example Derivation

Example

Want to determine the **semantics** of this command, using the **natural semantics** for the language that we just defined.

$$\overline{\text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \text{ -> } 7\}) \Downarrow \textbf{??}}$$

Example Derivation

Example

First, **if-then-else rule**, but we don't know if the guard is **true** or **false** yet.

$$\frac{}{\text{(if x > 5 then y := 2 + 3 else y := 3 + 4 fi, \{x -> 7\})} \Downarrow \textbf{??}}\ \textbf{If-??}$$

Example Derivation

Example

First, **if-then-else rule**, but we don't know if the guard is **true** or **false** yet.

$$\frac{(\mathbf{x > 5}, \{\mathbf{x \text{ -> } 7}\}) \Downarrow \mathbf{??}}{(\text{if } \mathbf{x > 5} \text{ then y := 2 + 3 else y := 3 + 4 fi}, \{\mathbf{x \text{ -> } 7}\}) \Downarrow \mathbf{??}} \ \textbf{If-??}$$

Example Derivation

Example

The guard is a **relation.**

$$\frac{(\mathbf{x > 5}, \mathbf{\{x \text{ -> } 7\}}) \Downarrow \mathbf{??}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \mathbf{??}} \; \textbf{If-??}$$

Example Derivation

Example

The guard is a **relation.**

$$\frac{\dfrac{(x, \{x\text{->}7\})\Downarrow \mathbf{??} \quad (5, \{x\text{->}7\})\Downarrow \mathbf{??} \quad \mathbf{??} > \mathbf{??} = \mathbf{??}}{(\mathbf{x > 5}, \mathbf{\{x \text{->} 7\}}) \Downarrow \mathbf{??}}\ \textbf{Rel}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \mathbf{??}}\ \textbf{If-??}$$

Example Derivation

Example

So we determine the meaning of **each side** of the **relation** …

$$\dfrac{\overline{(\mathbf{x}, \{x\text{->}7\})\Downarrow ??} \quad \overline{(\mathbf{5}, \{x\text{->}7\})\Downarrow ??} \quad ?? > ?? = ?? \;\; \textbf{Rel}}{\dfrac{(\mathbf{x} > \mathbf{5}, \{x \text{->} 7\}) \Downarrow ??}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow ??}\;\textbf{If-??}}$$

Example Derivation

Example

So we determine the meaning of **each side** of the **relation** …

$$\dfrac{\dfrac{\dfrac{}{(\mathbf{x}, \{x\text{->}7\}) \Downarrow \mathbf{7}}\,\text{Id} \qquad \dfrac{}{(\mathbf{5}, \{x\text{->}7\}) \Downarrow \mathbf{??}} \qquad \mathbf{7} > \mathbf{??} = \mathbf{??}}{(\mathbf{x} > \mathbf{5}, \{x \text{ -> } 7\}) \Downarrow \mathbf{??}}\,\textbf{Rel}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \ \{x \text{ -> } 7\}) \Downarrow \mathbf{??}}\,\textbf{If-??}$$

Example

So we determine the meaning of **each side** of the **relation** …

$$\dfrac{\dfrac{}{(\mathbf{x}, \{x\text{->}7\}) \Downarrow \mathbf{7}}\;\text{Id} \qquad \dfrac{}{(\mathbf{5}, \{x\text{->}7\}) \Downarrow \mathbf{5}}\;\text{Num} \qquad \mathbf{7} > \mathbf{5} = \mathbf{??}}{\dfrac{(\mathbf{x} > \mathbf{5}, \{x \text{->} 7\}) \Downarrow \mathbf{??}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \mathbf{??}}\;\textbf{If-??}}\;\textbf{Rel}$$

Example Derivation

Example

Then we use the **primitive** meaning of the **> relation**

$$\dfrac{\dfrac{}{(\mathbf{x}, \{x\text{->}7\}) \Downarrow \mathbf{7}}\,\textbf{Id} \qquad \dfrac{}{(\mathbf{5}, \{x\text{->}7\}) \Downarrow \mathbf{5}}\,\textbf{Num} \qquad \mathbf{7} > \mathbf{5} = \color{red}{\mathbf{??}}}{\dfrac{(\mathbf{x} > \mathbf{5}, \{x \text{ -> } 7\}) \Downarrow \color{red}{\mathbf{??}}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \color{red}{\mathbf{??}}}\,\textbf{If-}\color{red}{\mathbf{??}}}\,\textbf{Rel}$$

Example Derivation

Example

Then we use the **primitive** meaning of the **> relation**

$$\dfrac{\dfrac{\dfrac{}{(\mathbf{x}, \{x\text{->}7\})\Downarrow\mathbf{7}}\,\text{Id} \qquad \dfrac{}{(\mathbf{5}, \{x\text{->}7\})\Downarrow\mathbf{5}}\,\text{Num} \qquad \mathbf{7} > \mathbf{5} = \mathbf{true}}{(\mathbf{x} > \mathbf{5}, \{x \text{->} 7\}) \Downarrow \mathbf{??}}\,\text{Rel}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \; \{x \text{->} 7\}) \Downarrow \mathbf{??}}\,\text{If-??}$$

Example

Now, for the **if-then-else rule**, we know that the guard is **true**.

$$\dfrac{\dfrac{}{(x, \{x\text{->}7\})\Downarrow 7}\,\textbf{Id} \quad \dfrac{}{(5, \{x\text{->}7\})\Downarrow 5}\,\textbf{Num} \quad 7 > 5 = \textbf{true}}{\dfrac{(x > 5, \{x \text{->} 7\}) \Downarrow \textbf{??}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \textbf{??}}\,\textbf{If-??}}\,\textbf{Rel}$$

Example Derivation

Example

Now, for the **if-then-else rule**, we know that the guard is **true**.

$$\dfrac{\dfrac{\dfrac{}{(x, \{x\text{->}7\}) \Downarrow 7}\,\textbf{Id} \qquad \dfrac{}{(5, \{x\text{->}7\}) \Downarrow 5}\,\textbf{Num} \qquad 7 > 5 = \textbf{true}}{(x > 5, \{x \text{ -> } 7\}) \Downarrow \textbf{true}}\,\textbf{Rel}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \textbf{??}}\,\textbf{If-T}$$

Example

We are low on slide room, so let's squish what we're done with

$$\dfrac{\dfrac{}{(x, \{x\text{->}7\})\Downarrow 7}\,\text{Id} \qquad \dfrac{}{(5, \{x\text{->}7\})\Downarrow 5}\,\text{Num} \qquad 7 > 5 = \mathbf{true}}{\dfrac{(x > 5, \{x \text{->} 7\}) \Downarrow \mathbf{true}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \mathbf{??}}\,\textbf{If-T}}\,\textbf{Rel}$$

Example Derivation

Example

We are low on slide room, so let's squish what we're done with

$$\dfrac{\dfrac{\dots}{(x > 5, \{x \text{ -> } 7\}) \Downarrow \text{true}}\ \text{Rel}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \textbf{??}}\ \text{If-T}$$

Example Derivation

Example

Now what?

$$\dfrac{\dfrac{\ldots}{(x > 5, \{x \text{ -> } 7\}) \Downarrow \text{true}}\;\textbf{Rel}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \textbf{??}}\;\textbf{If-T}$$

Example

Now what?

$$\dfrac{\dfrac{\ldots}{(x > 5, \{x \text{ -> } 7\}) \Downarrow \text{true}}\ \textbf{Rel}}{(\text{if } x > 5 \text{ then } \mathbf{y := 2 + 3} \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \textbf{??}}\ \textbf{If-T}$$

Example

We need the meaning of the **if** branch, not the **else** branch

$$\dfrac{\dfrac{\ldots}{(x > 5, \{x \text{ -> } 7\}) \Downarrow \text{true}}\,\textbf{Rel} \qquad \begin{array}{c}(\mathbf{y := 2 + 3}, \{x \text{ -> } 7\})\\ \Downarrow \ ??\end{array}}{(\text{if } x > 5 \text{ then } \mathbf{y := 2 + 3} \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \ ??}\,\textbf{If-T}$$

Example Derivation

Example

This is an **assignment**

$$\dfrac{\dfrac{\dots}{(x > 5, \{x \text{ -> } 7\}) \Downarrow \text{true}}\,\text{Rel} \qquad \dfrac{(\mathbf{2+3}, \{x\text{->}7\}) \Downarrow \mathbf{??}}{(y := \mathbf{2 + 3}, \{x \text{ -> } 7\})}\,\text{Assign} \; \Downarrow \mathbf{??}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \mathbf{??}}\,\text{If-T}$$

Example Derivation

Example

The body is an **arithmetic** expression

$$\dfrac{\dfrac{\ldots}{(x > 5, \{x \text{ -> } 7\}) \Downarrow \text{true}}\,\textbf{Rel} \qquad \dfrac{\dfrac{(\mathbf{2}, \{x\text{->}7\}) \Downarrow \mathbf{??} \quad (\mathbf{3}, \{x\text{->}7\}) \Downarrow \mathbf{??} \quad \mathbf{??} + \mathbf{??} = \mathbf{??}}{(\mathbf{2+3}, \{x\text{->}7\}) \Downarrow \mathbf{??}}\,\textbf{Arith}}{(y := 2 + 3, \{x \text{ -> } 7\})}\,\textbf{Assign} \;\Downarrow \mathbf{??}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \mathbf{??}}\,\textbf{If-T}$$

Example Derivation

Example

Determine meaning of **each side**

Num

(**2**, {x->7})⇓**2**

Num

(**3**, {x->7})⇓**3**

2 + **3** = **??**

Arith

(**2**+**3**, {x->7})⇓**??**

Assign

(y := 2 + 3, {x -> 7})

...

Rel

(x > 5, {x -> 7}) ⇓ true

⇓ **??**

If-T

(if x > 5 then y := 2 + 3 else y := 3 + 4 fi,

{x -> 7}) ⇓ **??**

Example Derivation

Example

Then use the **primitive** meaning of the operation

$$\dfrac{\dfrac{\dfrac{}{(2, \{x\text{->}7\})\Downarrow 2}\,\textbf{Num} \quad \dfrac{}{(3, \{x\text{->}7\})\Downarrow 3}\,\textbf{Num} \quad \mathbf{2} + \mathbf{3} = \mathbf{5}}{(2+3, \{x\text{->}7\})\Downarrow \textcolor{red}{\mathbf{??}}}\,\textbf{Arith}}{(y := 2 + 3, \{x \text{->} 7\})}\,\textbf{Assign}$$

$$\dfrac{\dfrac{\dots}{(x > 5, \{x \text{->} 7\}) \Downarrow \text{true}}\,\textbf{Rel} \qquad \Downarrow \textcolor{red}{\mathbf{??}}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \textcolor{red}{\mathbf{??}}}\,\textbf{If-T}$$

Example Derivation

Example

We can now fill in the remaining details

$$\dfrac{\dfrac{\dfrac{}{(2, \{x\text{->}7\})\Downarrow 2}\,\text{Num} \quad \dfrac{}{(3, \{x\text{->}7\})\Downarrow 3}\,\text{Num} \quad 2 + 3 = \mathbf{5}}{(2+3, \{x\text{->}7\})\Downarrow \color{red}{\mathbf{??}}}\,\text{Arith}}{(y := 2 + 3, \{x \text{->} 7\})}\,\text{Assign}$$

$$\dfrac{\dfrac{\mathbf{...}}{(x > 5, \{x \text{->} 7\}) \Downarrow \text{true}}\,\text{Rel} \qquad \Downarrow \color{red}{\mathbf{??}}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \color{red}{\mathbf{??}}}\,\text{If-T}$$

Example Derivation

Example

We can now fill in the remaining details

$$
\dfrac{
\dfrac{\dots}{(x > 5, \{x \text{ -> } 7\}) \Downarrow \text{true}}\,\textbf{Rel}
\qquad
\dfrac{
\dfrac{
\dfrac{}{(2, \{x\text{->}7\})\Downarrow 2}\,\textbf{Num}
\quad
\dfrac{}{(3, \{x\text{->}7\})\Downarrow 3}\,\textbf{Num}
\quad
2 + 3 = \mathbf{5}
}{(2+3, \{x\text{->}7\})\Downarrow \mathbf{5}}\,\textbf{Arith}
}{(y := 2 + 3, \{x \text{ -> } 7\}) \Downarrow \color{red}{\mathbf{??}}}\,\textbf{Assign}
}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow \color{red}{\mathbf{??}}}\,\textbf{If-T}
$$

Example Derivation

Example

We can now fill in the remaining details

$$\dfrac{\dfrac{\cdots}{(x > 5, \{x \text{->} 7\}) \Downarrow \text{true}}\,\text{Rel} \qquad \dfrac{\dfrac{\dfrac{}{(2, \{x\text{->}7\})\Downarrow 2}\,\text{Num} \quad \dfrac{}{(3, \{x\text{->}7\})\Downarrow 3}\,\text{Num} \quad 2 + 3 = \mathbf{5}}{(2{+}3, \{x\text{->}7\})\Downarrow \mathbf{5}}\,\text{Arith}}{(y := 2 + 3, \{x \text{->} 7\}) \Downarrow \mathbf{\{x\text{->}7, y\text{->}5\}}}\,\text{Assign}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \mathbf{??}}\,\text{If-T}$$

Example

We can now fill in the remaining details

$$
\dfrac{\dfrac{\dfrac{\dots}{(x > 5, \{x \to 7\}) \Downarrow \text{true}}\,\text{Rel} \qquad \dfrac{\dfrac{\dfrac{}{(2, \{x\text{->}7\}) \Downarrow 2}\,\text{Num} \quad \dfrac{}{(3, \{x\text{->}7\}) \Downarrow 3}\,\text{Num} \quad 2 + 3 = \mathbf{5}}{(2+3, \{x\text{->}7\}) \Downarrow \mathbf{5}}\,\text{Arith}}{(y := 2 + 3, \{x \text{->} 7\}) \Downarrow \mathbf{\{x\text{->}7, y\text{->}5\}}}\,\text{Assign}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \mathbf{\{x\text{->}7, y\text{->}5\}}}\,\text{If-T}
$$

Example Derivation

Questions so far?

Awkward Example

Let in Command

$\boxed{(C, m) \Downarrow m'}$

$$\frac{(E, m) \Downarrow v \quad (C, m[I<\!-v]) \Downarrow m'}{(\text{let } I = E \text{ in } C, m) \Downarrow m''}$$

Where $m''(y) = m'(y)$ for $y \neq I$ and
$m''(I) = m(I)$ if $m(I)$ is defined,
and $m''(I)$ is undefined otherwise

Awkward Example

Let in Command

$$\dfrac{\texttt{(5,\{x->17\})} \Downarrow 5 \qquad \dfrac{\dfrac{\texttt{(x,\{x->5\})} \Downarrow 5 \quad \texttt{(3,\{x->5\})} \Downarrow 3}{\texttt{(x+3,\{x->5\})} \Downarrow 8}}{\texttt{(x:=x+3,\{x->5\})} \Downarrow \texttt{\{x->8\}}}}{\texttt{(let x = 5 in (x:=x+3), \{x -> 17\})} \Downarrow \textbf{??}}$$

Awkward Example

Let in Command

$$\frac{(5,\{x\text{->}17\}) \Downarrow 5 \qquad \dfrac{\dfrac{(x,\{x\text{->}5\}) \Downarrow 5 \quad (3,\{x\text{->}5\}) \Downarrow 3}{(x+3,\{x\text{->}5\}) \Downarrow 8}}{(x:=x+3,\{x\text{->}5\}) \Downarrow \{x\text{->}8\}}}{(\text{let } x = 5 \text{ in } (x:=x+3),\ \{x \text{ -> } 17\}) \Downarrow \boldsymbol{\{x\text{->}17\}}}$$

Awkward Example

Comment

- Simple Imperative Programming Language introduces variables **implicitly** through assignment
- The let-in command introduces scoped variables **explictly**
- **Clash** of constructs apparent in **awkward** semantics

Awkward Example

Questions so far?

Implementing Semantics

Interpretation Versus Compilation

- A **compiler** from language L1 to language L2 is a program that takes an L1 program and for each piece of code in L1 **generates** a piece of **code** in L2 of **same meaning**
- An **interpreter** of L1 in L2 is an L2 program that **executes the meaning** of a given L1 program
- Compiler would examine the body of a loop once; an interpreter would examine it every time the loop was executed

Interpretation Versus Compilation

- A **compiler** from language L1 to language L2 is a program that takes an L1 program and for each piece of code in L1 **generates** a piece of **code** in L2 of **same meaning**
- An **interpreter** of L1 in L2 is an L2 program that **executes the meaning** of a given L1 program
- Compiler would examine the body of a loop once; an interpreter would examine it every time the loop was executed

Interpreter

- An **Interpreter** represents the **operational semantics** of a language L1 (**source** language) in the language of implementation L2 (**target** language)
- Built incrementally
  - Start with literals
  - Variables
  - Primitive operations
  - Evaluation of expressions
  - Evaluation of commands/declarations

Interpreter

- Takes abstract syntax trees as input
  - In simple cases could be just strings
- One procedure for each syntactic category (nonterminal)
  - e.g., one for expressions, another for commands
- From semantics to implementation:
  - If Natural Semantics used, tells how to compute final value from code
  - If Transition Semantics used, tells how to compute next "state"
    - To get final value, put in a loop

Interpreter

- Takes abstract syntax trees as input
  - In simple cases could be just strings
- One procedure for each syntactic category (nonterminal)
  - e.g., one for expressions, another for commands
- From semantics to implementation:
  - If Natural Semantics used, tells how to compute final value from code
  - If Transition Semantics used, tells how to compute next "state"
    - To get final value, put in a loop

Natural Semantics Example

- compute_exp (Var(v), m) = look_up v m
- compute_exp (Int(n), _) = Num (n)
- …
- compute_com(IfExp(b,c1,c2),m) =

  if compute_exp (b,m) = Bool(true)

  then compute_com (c1,m)

  else compute_com (c2,m)

Natural Semantics Example

- compute_com(While(b,c), m) =
  if compute_exp (b,m) = Bool(false)
  then m
  else compute_com
  (While(b,c), compute_com(c,m))

- May fail to terminate - exceed stack limits
- Returns no useful information then

Questions?

No Class Thursday for Midterm!