



Today's Cunning Plan

- Review, Truth, and Provability
- Large-Step Opsem Commentary
- Small-Step Contextual Semantics
 - Reductions, Redexes, and Contexts
- Applications and Recent Research

60 Second Summary: Semantics

- A formal semantics is a system for assigning meanings to programs.
- For now, programs are IMP commands and expressions
- In operational semantics the meaning of a program is “what it evaluates to”
- Any opsem system gives rules of inference that tell you how to evaluate programs

Summary - Judgments

- Rules of inference allow you to derive judgments (“something that is knowable”) like

$$\langle e, \sigma \rangle \Downarrow n$$

- In state σ , expression e evaluates to n

$$\langle c, \sigma \rangle \Downarrow \sigma'$$

- After evaluating command c in state σ the new state will be σ'

- State σ maps variables to values ($\sigma : L \rightarrow Z$)
- Inferences equivalent up to variable renaming:

$$\langle c, \sigma \rangle \Downarrow \sigma' \quad === \quad \langle c', \sigma_7 \rangle \Downarrow \sigma_8$$

Notation: Rules of Inference

- We express the evaluation rules as rules of inference for our judgment
 - called the derivation rules for the judgment
 - also called the evaluation rules (for operational semantics)
- In general, we have **one rule for each language construct**:

$$\frac{\langle e_1, \sigma \rangle \Downarrow n_1 \quad \langle e_2, \sigma \rangle \Downarrow n_2}{\langle e_1 + e_2, \sigma \rangle \Downarrow n_1 + n_2}$$

This is the only rule for $e_1 + e_2$

Evaluation By Inversion

- We must find n_1 and n_2 such that $e_1 \Downarrow n_1$ and $e_2 \Downarrow n_2$ are derivable
 - This is done **recursively**
- If there is exactly one rule for each kind of expression we say that the rules are syntax-directed
 - At each step at most one rule applies
 - This allows a simple evaluation procedure as above (recursive tree-walk)
 - True for our Aexp but not Bexp.

Summary - Rules

- Rules of inference list the hypotheses necessary to arrive at a conclusion

$$\begin{array}{c}
 \text{_____} \\
 \langle x, \sigma \rangle \Downarrow \sigma(x)
 \end{array}
 \qquad
 \begin{array}{c}
 \langle e_1, \sigma \rangle \Downarrow n_1 \quad \langle e_2, \sigma \rangle \Downarrow n_2 \\
 \text{_____} \\
 \langle e_1 - e_2, \sigma \rangle \Downarrow n_1 \text{ minus } n_2
 \end{array}$$

- A derivation involves interlocking (well-formed) instances of rules of inference

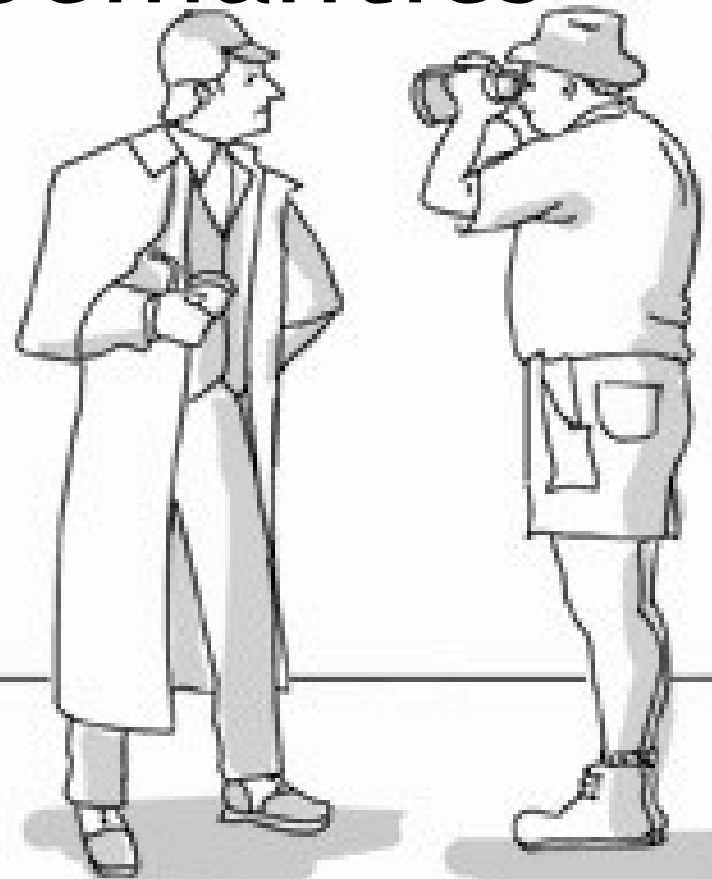
$$\begin{array}{c}
 \langle 4, \sigma_3 \rangle \Downarrow 4 \quad \langle 2, \sigma_3 \rangle \Downarrow 2 \\
 \text{_____} \\
 \langle 4 * 2, \sigma_3 \rangle \Downarrow 8 \qquad \langle 6, \sigma_3 \rangle \Downarrow 6 \\
 \text{_____} \\
 \langle (4 * 2) - 6, \sigma_3 \rangle \Downarrow 2
 \end{array}$$

Operational Semantics

Small-Step Semantics



Sherlock saw the man using binoculars.



Sherlock saw the man using binoculars.

Provability



- Given an opsem system, $\langle e, \sigma \rangle \Downarrow n$ is provable *if there exists* a well-formed derivation with $\langle e, \sigma \rangle \Downarrow n$ as its conclusion
 - “well-formed” = “every step in the derivation is a valid instance of one of the rules of inference for this opsem system”
 - “ $\vdash \langle e, \sigma \rangle \Downarrow n$ ” = “it is provable that $\langle e, \sigma \rangle \Downarrow n$ ”
- We would *like* truth and provability to be closely related



Truth?



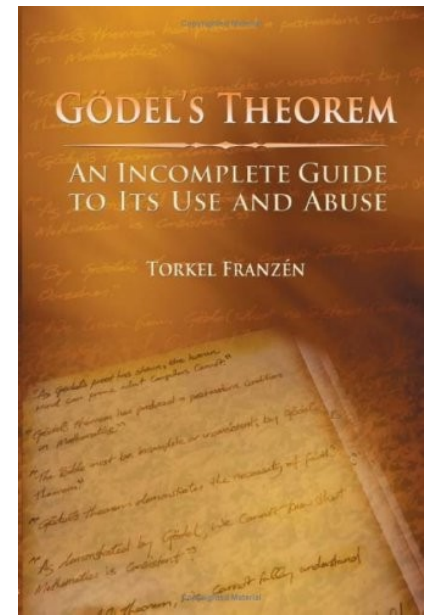
- We will **not** formally define “**truth**” yet
- Instead we appeal to your **intuition**
 - $\langle 2+2, \sigma \rangle \Downarrow 4$ -- *should be* true
 - $\langle 2+2, \sigma \rangle \Downarrow 5$ -- *should be* false
- (See Tarski's semantic theory of truth. Informally, “some cats are fluffy” is true if and only if some cats are, actually, fluffy.)

Completeness

- A proof system (like our operational semantics) is complete if every true judgment is provable.
- If we *replaced* the subtract rule with:

$$\frac{\langle e_1, \sigma \rangle \Downarrow n \quad \langle e_2, \sigma \rangle \Downarrow 0}{\langle e_1 - e_2, \sigma \rangle \Downarrow n}$$

- Our opsem would be incomplete:
 $\langle 4-2, \sigma \rangle \Downarrow 2$ -- true but not provable



Consistency

- A proof system is consistent (or sound) if every provable judgment is true.
- If we **replaced** the subtract rule with:

$$\frac{\langle e_1, \sigma \rangle \Downarrow n_1 \quad \langle e_2, \sigma \rangle \Downarrow n_2}{\langle e_1 - e_2, \sigma \rangle \Downarrow n_1 + 3}$$

- Our opsem would be inconsistent (or unsound):
 - $\langle 6-1, \sigma \rangle \Downarrow 9$ -- false but provable

“A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines.”
-- Ralph Waldo Emerson, *Essays. First Series. Self-Reliance.*

Desired Traits

- Typically a system (of operational semantics) is always **complete** (unless you forget a rule)
- Without care, your system may be **unsound**
- Usually that is **very bad**
 - A paper with an unsound type system is usually rejected
 - Papers often prove (sketch) that a system is sound
 - Much classic and recent (e.g., from Engler or ESP to ChatGPT) work covers useful but unsound systems
- In this class **your work should be complete and consistent** (e.g., on homework problems)

Dr. Peter Venkman: I'm a little fuzzy on the whole "good/bad" thing here. What do you mean, "bad"?

Dr. Egon Spengler: Try to imagine all life as you know it stopping instantaneously and every molecule in your body exploding at the speed of light.

With That In Mind

- We now return to opsem for IMP

$$\frac{\langle e, \sigma \rangle \Downarrow n}{\langle x := e, \sigma \rangle \Downarrow \sigma[x := n]}$$

Def: $\sigma[x := n](x) = n$ $\sigma[x := n](y) = \sigma(y)$

$$\frac{\langle b, \sigma \rangle \Downarrow \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \text{true} \quad \langle c; \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'}{\langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'}$$

Command Evaluation Notes

- The order of evaluation is important
 - c_1 is evaluated **before** c_2 in $c_1; c_2$
 - c_2 is **not** evaluated in “if true then c_1 else c_2 ”
 - c is **not** evaluated in “while false do c ”
 - b is evaluated **first** in “if b then c_1 else c_2 ”
 - this is explicit in the evaluation rules
- Conditional constructs (e.g., $b_1 \vee b_2$) have multiple evaluation rules
 - but only one can be applied at one time

Command Evaluation Trials

- The evaluation rules are not syntax-directed
 - See the rules for **while**, **^**
 - The evaluation **might not terminate**
- Recall: the evaluation rules suggest an interpreter
- Natural-style semantics has two big disadvantages (continued ...)

Disadvantages of Natural-Style Operational Semantics

- It is hard to talk about commands whose evaluation does **not terminate**
 - When there is **no** σ' such that $\langle c, \sigma \rangle \Downarrow \sigma'$
 - But that is true also of ill-formed or erroneous commands (in a richer language)!
- It does not give us a way to talk about **intermediate states**
 - Thus we cannot say that on a parallel machine the execution of two commands is interleaved (= **no modeling threads**)

Semantics Solution



- Small-step semantics addresses these problems
 - Execution is modeled as a (possible infinite) **sequence of states**
- Not quite as easy as large-step natural semantics, though
- Contextual semantics is a small-step semantics where the atomic execution step is a rewrite of the program

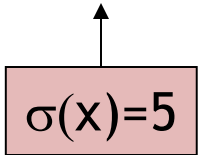
Contextual Semantics

- We will define a relation $\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle$
 - c' is obtained from c via an **atomic rewrite step**
 - Evaluation terminates when the program has been rewritten to a **terminal program**
 - one from which we cannot make further progress
 - For IMP the terminal command is “skip”
 - As long as the command is not “skip” we can make further progress
 - some commands *never* reduce to skip (e.g., “while true do skip”)

Contextual Derivations

- In small-step contextual semantics, derivations are not tree-structured
- A contextual semantics derivation is a sequence (or list) of atomic rewrites:

$$\langle x+(7-3), \sigma \rangle \rightarrow \langle x+(4), \sigma \rangle \rightarrow \langle 5+4, \sigma \rangle \rightarrow \langle 9, \sigma \rangle$$



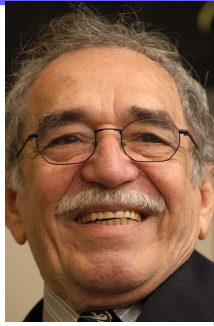
$\sigma(x)=5$

What is an Atomic Reduction?

- What is an atomic reduction step?
 - Granularity is a choice of the semantics designer
- How to select the next reduction step, when several are possible?
 - This is the **order of evaluation** issue



Columbian Spanish Literature



- This Colombian novelist received the Nobel Prize for Literature and is viewed as one of the most significant authors in the 20th century. His works include *Cien años de soledad*, *Crónica de una muerte anunciada* and *El amor en los tiempos del cólera*. He helped popularize the magical realism literary style.
- Bonus: What is Macondo?

Correcting English Prose

4. Lizzy drank in the sight of him like a thirst craven man consumes water.

421. "I go here, silly," said Kimi with a proud expression. "And how I might ask? Your scores were not legible for this school."

312. Every member of the Thespians, or anyone who has ever acted in one of our school plays was a pre-Madonna, mellow-dramatic; over-actor and I didn't want to be one of them.

198. Nobody goes into Donovan's Layer, For they sence evil. But Livvy doesn't she see's something no one else does.

Geography



- The city of Konya is the sixth-largest in *this country*. It houses a tropical butterfly garden and historical mosques. It has been ruled by Cimmerians, Persians, Pergamon kings, and the Roman and Ottoman empires. Legend has it that its earlier name Ikónion refers to the Greek legend of Perseus using the Gorgon Medusa's head before founding the city.

Sports

- This NFL Football Team is associated with Brett Favre, Mike McCarthy and Matt LaFleur. The team is the only community-owned franchise among the NBA, NFL, NHL and MLB. It is owned by over 500,000 stockholders and no one is allowed to hold more than 200,000 shares. As a publicly-held nonprofit, this team is the only American major-league sports franchise to release financial records each year.



Q: Computer Science

- This American computer scientist won the 2008 Turing award for her work on design of programming languages and software methodology that led to the development of object-oriented programming. In addition to the first high-level language to support distributed programs and notable results on Byzantine fault tolerance, she is perhaps best known for her formulation of object-oriented subtyping.
- Bonus: What is her eponymous principle?

Redexes

- A redex is a syntactic expression or command that **can be reduced** (transformed) **in one atomic step**
- Redexes are defined via a grammar:

$r ::= x \qquad (x \in L)$

| $n_1 + n_2$

| $x := n$

| skip; c

| if true then c_1 else c_2

| if false then c_1 else c_2

| while b do c

- For brevity, we mix exp and command redexes
- Note that $(1 + 3) + 2$ is **not** a redex, but $1 + 3$ is

Local Reduction Rules for IMP

- One for each redex: $\langle r, \sigma \rangle \rightarrow \langle e, \sigma' \rangle$
 - means that in state σ , the redex r can be *replaced in one step* with the expression e

$$\langle x, \sigma \rangle \rightarrow \langle \sigma(x), \sigma \rangle$$

$$\langle n_1 + n_2, \sigma \rangle \rightarrow \langle n, \sigma \rangle \quad \text{where } n = n_1 \text{ plus } n_2$$

$$\langle n_1 = n_2, \sigma \rangle \rightarrow \langle \text{true}, \sigma \rangle \quad \text{if } n_1 = n_2$$

$$\langle x := n, \sigma \rangle \rightarrow \langle \text{skip}, \sigma[x := n] \rangle$$

$$\langle \text{skip}; c, \sigma \rangle \rightarrow \langle c, \sigma \rangle$$

$$\langle \text{if true then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \langle c_1, \sigma \rangle$$

$$\langle \text{if false then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \langle c_2, \sigma \rangle$$

$$\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow$$

$$\langle \text{if } b \text{ then } c; \text{ while } b \text{ do } c \text{ else skip}, \sigma \rangle$$

The Global Reduction Rule

- General idea of contextual semantics
 - **Decompose** the current expression into the **redex**-to-reduce-next and the remaining program
 - The remaining program is called a context
 - Reduce the redex “r” to some other expression “e”
 - The resulting (reduced) expression consists of “e” with the original context

As A Picture (1)

(Context)

...

$x := 2+2 ;$

print x

Step 1: Find The Redex

As A Picture (2)

(Context)

...

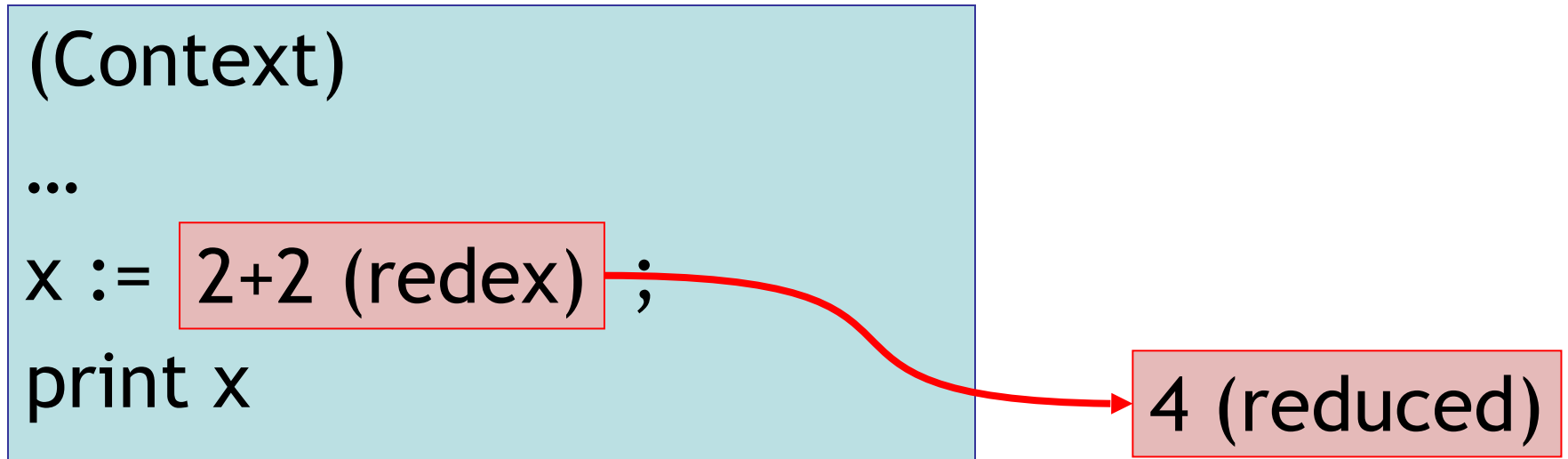
x := 2+2 (redex) ;

print x

Step 1: Find The Redex

Step 2: Reduce The Redex

As A Picture (3)



Step 1: Find The Redex

Step 2: Reduce The Redex

As A Picture (4)

(Context)

...

x := 4 ;

print x

Step 1: Find The Redex

Step 2: Reduce The Redex

Step 3: Replace It In The Context

Contextual Analysis

- We use H to range over **contexts**
- We write $H[r]$ for the expression obtained by placing redex r in context H
- Now we can define a small step

If $\langle r, \sigma \rangle \rightarrow \langle e, \sigma' \rangle$

then $\langle H[r], \sigma \rangle \rightarrow \langle H[e], \sigma' \rangle$

Contexts

- A context is like an expression (or command) with a marker \bullet in the place where the **redex** goes
- Examples:
 - To evaluate “ $(1 + 3) + 2$ ” we use the redex **$1 + 3$** and the context “ $\bullet + 2$ ”
 - To evaluate “if $x > 2$ then c_1 else c_2 ” we use the redex **x** and the context “if $\bullet > 2$ then c_1 else c_2 ”

Context Terminology

- A context is also called an “expression with a hole”
- The marker • is sometimes called a hole
- $H[r]$ is the expression obtained from H by replacing • with the redex r

“Avoid context and specifics; generalize and keep repeating the generalization.”
-- Jack Schwartz

Contextual Semantics Example

- $x := 1 ; x := x + 1$ with initial state $[x:=0]$

<Comm, State>	Redex •	Context
<x := 1; x := x+1, [x := 0]>	$x := 1$	$\bullet; x := x+1$
<skip; x := x+1, [x := 1]>	$\text{skip}; x := x+1$	\bullet
<x := x+1, [x := 1]>	x	$x := \bullet + 1$
What happens next?		

Contextual Semantics Example

- $x := 1 ; x := x + 1$ with initial state $[x:=0]$

<Comm, State>	Redex •	Context
<x := 1; x := x+1, [x := 0]>	$x := 1$	$\bullet; x := x+1$
<skip; x := x+1, [x := 1]>	$\text{skip}; x := x+1$	\bullet
<x := x+1, [x := 1]>	x	$x := \bullet + 1$
<x := 1 + 1, [x := 1]>	$1 + 1$	$x := \bullet$
<x := 2, [x := 1]>	$x := 2$	\bullet
<skip, [x := 2]>		

More On Contexts

- **Contexts** are defined by a grammar:

$$\begin{aligned} H ::= & \bullet \mid n + H \\ & \mid H + e \\ & \mid x := H \\ & \mid \text{if } H \text{ then } c_1 \text{ else } c_2 \\ & \mid H; c \end{aligned}$$

- A context has **exactly one** \bullet marker
- A redex is never a value

What's In A Context?

- Contexts specify precisely how to find the next redex
 - Consider $e_1 + e_2$ and its decomposition as $H[r]$
 - If e_1 is n_1 and e_2 is n_2 then $H = \bullet$ and $r = n_1 + n_2$
 - If e_1 is n_1 and e_2 is not n_2 then $H = n_1 + H_2$ and $e_2 = H_2[r]$
 - If e_1 is not n_1 then $H = H_1 + e_2$ and $e_1 = H_1[r]$
 - In the last two cases the decomposition is done recursively
 - Check that in each case the solution is unique

Unique Next Redex:

Proof By Handwaving Examples

- Suppose $c = “c_1; c_2”$. Then **either**
 - $c_1 = \text{skip}$ and then $c = H[\text{skip}; c_2]$ with $H = \bullet$
 - **or** $c_1 \neq \text{skip}$ and then $c_1 = H[r]$; so $c = H'[r]$ with $H' = H; c_2$
- Suppose $c = “\text{if } b \text{ then } c_1 \text{ else } c_2”$. Then
 - **either** $b = \text{true}$ or $b = \text{false}$ and then $c = H[r]$ with $H = \bullet$
 - **or** b is not a value and $b = H[r]$; so $c = H'[r]$ with $H' = \text{if } H \text{ then } c_1 \text{ else } c_2$

Context Decomposition

- Decomposition theorem:

If **c** is not “skip” then there exist unique **H** and **r** such that **c** is **H[r]**

- “Exist” means progress
- “Unique” means determinism



Short-Circuit Evaluation

- What if we want to express **short-circuit** evaluation of \wedge ?
 - Define the following **contexts**, **redexes** and **local reduction rules**
$$H ::= \dots \mid H \wedge b_2$$
$$r ::= \dots \mid \text{true} \wedge b \mid \text{false} \wedge b$$
$$\langle \text{true} \wedge b, \sigma \rangle \rightarrow \langle b, \sigma \rangle$$
$$\langle \text{false} \wedge b, \sigma \rangle \rightarrow \langle \text{false}, \sigma \rangle$$
 - the local reduction kicks in **before** b_2 is **evaluated**

Contextual Semantics Summary

- Can view • as representing the **program counter**
- The advancement rules for • are non-trivial
 - At each step the **entire command** is decomposed
 - This makes contextual semantics **inefficient to implement directly**
- The major advantage of contextual semantics: it allows a *mix* of local and global reduction rules
 - For IMP we have only local reduction rules: only the redex is reduced
 - Sometimes it is useful to work on the context too
 - We'll do that when we study **memory allocation**, etc.

Reading Real-World Examples

- Cobbe and Felleisen, POPL 2005
- Small-step contextual opsem for Java
- Their rule for object field access:

$$\boxed{P \vdash \langle E[obj.fd], S \rangle \hookrightarrow \langle E[\mathcal{F}(fd)], S \rangle}$$

where $\mathcal{F} = \text{fields}(S(obj))$ and $fd \in \text{dom}(\mathcal{F})$

$$P \vdash \langle E[obj.fd], S \rangle \rightarrow \langle E[F(fd)], S \rangle$$

- where $F = \text{fields}(S(obj))$ and $fd \in \text{dom}(F)$

- They use “E” for context, we use “H”
- They use “S” for state, we use “ σ ”

Lost In Translation

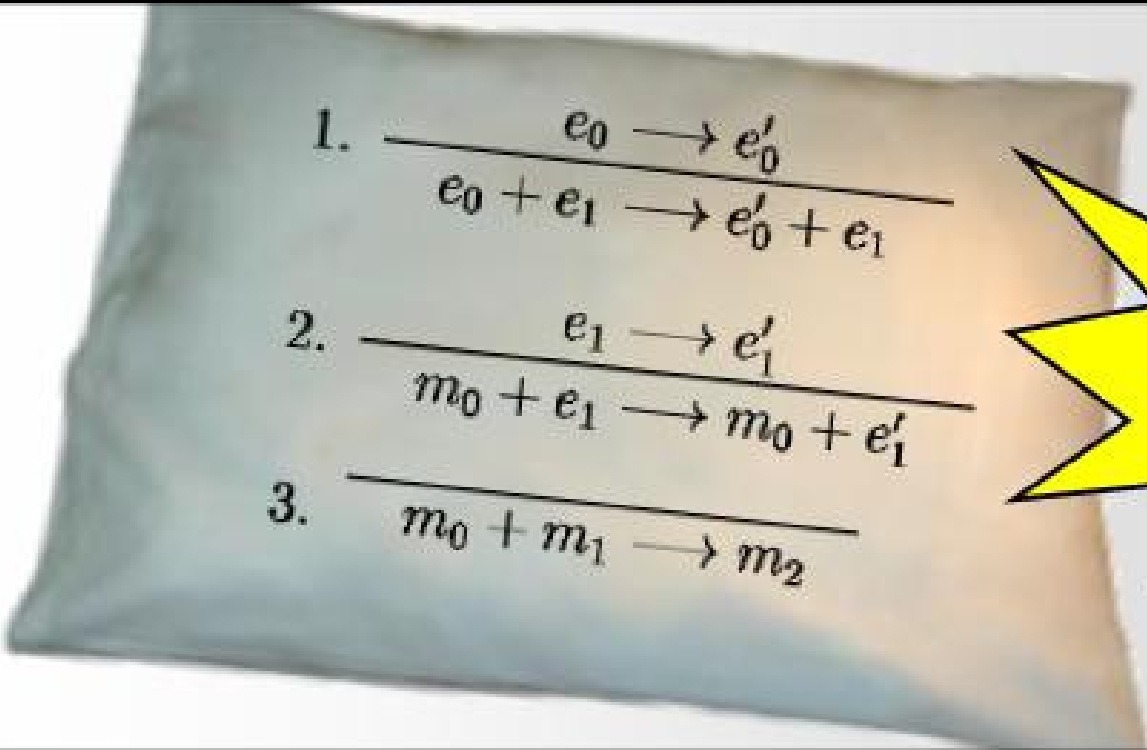
- $P \vdash \langle H[\text{obj.fd}], \sigma \rangle \rightarrow \langle H[F(\text{fd})], \sigma \rangle$
 - Where $F = \text{fields}(\sigma(\text{obj}))$ and $\text{fd} \in \text{dom}(F)$
- They have “ $P \vdash$ ”, but that just means “it can be proved in our system given P ”
- $\langle H[\text{obj.fd}], \sigma \rangle \rightarrow \langle H[F(\text{fd})], \sigma \rangle$
 - Where $F = \text{fields}(\sigma(\text{obj}))$ and $\text{fd} \in \text{dom}(F)$

Lost In Translation 2

- $\langle H[\text{obj}.fd], \sigma \rangle \rightarrow \langle H[F(fd)], \sigma \rangle$
 - Where $F = \text{fields}(\sigma(\text{obj}))$ and $fd \in \text{dom}(F)$
- They model objects (like **obj**), but we do not (yet) - let's just make fd a variable:
- $\langle H[fd], \sigma \rangle \rightarrow \langle H[F(fd)], \sigma \rangle$
 - Where $F = \sigma$ and $fd \in L$
- Which is just our variable-lookup rule:
- $\langle H[fd], \sigma \rangle \rightarrow \langle H[\sigma(fd)], \sigma \rangle$ (when $fd \in L$)

“Sleep On It”

“The Semantics Pillow”

- 
1.
$$\frac{e_0 \rightarrow e'_0}{e_0 + e_1 \rightarrow e'_0 + e_1}$$
 2.
$$\frac{e_1 \rightarrow e'_1}{m_0 + e_1 \rightarrow m_0 + e'_1}$$
 3.
$$\frac{}{m_0 + m_1 \rightarrow m_2}$$

**Only
\$19,95**

“Learn while you sleep!”

Homework

- Homework 1 Due soon
- Reading!