6.044 Fall 1991

9/11/1991

Prof. Albert Meyer
TA: Arthur Lent

2 ~~Hand~~

Handouts
1. Course info Sheet
2. Diagnostic Quiz (due Friday) (make more)
3A. Sign-up sheet (sign and pass on)

students = 24 at start.

Goal of class - software systems

additional agenda - rudiments of Mathematical Logic (20%)
- general theory of computability (25%)

The Text INTRODUCTION TO THE FORMAL SEMANTICS OF PROGRAMMING LANGUAGES
- Glynn Winskel

<u>Mail List.</u>

Quizzes v. Final Exams

Formal definition of a programming language of "while-programs"; IMP (Imperative)
$X := 3$

BNF Backus - Naur Form

arithmetic expressions: <u>Aexp</u>
boolean " : <u>Bexp</u>
Commands : <u>Com</u>

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

"an arithmetic expression is defined inductively as follows it could be a numeral, a location, a sum of 2 arithmetic expressions ..."

we will be safely but usefully vague about parentheses

$$3 + X + 9 \times 7$$

(( ))

Num n, m

Loc A, X, B, Y

Bexp $b ::= \underline{true} \mid \underline{false} \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid a_1 = a_2 \mid a_1 \leq a_2$

Com $c ::= \underline{skip} \mid X := a \mid c_0 ; c_1 \mid \underline{if}\ b\ \underline{then}\ c_0\ \underline{else}\ c_1 \mid \underline{while}\ b\ \underline{do}\ c$

---

p. 32 Euclid:

$\underline{while}\ \neg(M = N)\ \underline{do}$

$\quad \underline{if}\ M \leq N$

$\quad \underline{then}\ N := N - M$

$\quad \underline{else}\ M := M - N$

if M, N nonnegative ints it always terminates with GCD in N and M.

This is the sort of fact we want in this course.

CLAIM: if M, N are initially positive integers then Euclid will halt with final values of M, N, both equal to the gcd of the original values.

6044S91
p2
9/11

kinds of semantics:

Operational semantics:
1. "evaluates to" relation, inductive.
2. "rewriting style"

denotational semantics: assigns a mathematical "meaning" such as a number, or a function of numbers, or a functional on ... to commands.

axiomatic semantics:

Styles of explaining how programs behave and what they mean.

later we will look at recursive functional programs.
(recursive types?) maybe

Concurrent programs.

undecidability of halting problem

incompleteness in logic

$M', N' \notin loc(Euclid)$

while $M \neq N$ do

$\underbrace{\{M' = M \text{ and } N' = N \text{ and } M, N > 0\}}_{P}$ Euclid $\underbrace{\{M = gcd(M', N')\}}_{Q}$

Proof: Find an assertion A s.t. ← loop invariant

(1) P implies A

(2) $A \wedge M = N$ implies Q

(3) $\{A \text{ and } M \neq N\}$ if $M \leq N$ then $N := N - M$ else $M := M - N$ $\{A\}$

"Magic": Let A be

$$M, N > 0 \text{ and } gcd(M', N') = gcd(M, N)$$

(1) obvious (2) follows because if $M = N$ (>0), then $gcd(M, N) = M$.

to prove (3), find B s.t.

(4) A and $M \neq N$ implies B

(5) $\{B \text{ and } M \leq N\}$ ~~implies~~ $N := N - M$ $\{A\}$

(6) $\{B \text{ and } M > N\}$ $M := M - N$ $\{A\}$

~~"Semi-automatic "Magic"~~ Let B be

$$(M \leq N \text{ and } A[^{N-M}/_N]) \text{ or } (M > N \text{ and } A[^{M-N}/_N])$$

Proof of (4) Say $M, N > 0$ and $gcd(M', N') = gcd(M, N)$ and $M \neq N$.

Two cases $0 < M < N$ and $0 < N < M$

Case 1. $0 < M \leq N$:

Then $gcd(M, N) = gcd(M, N-M)$

by elementary number theory: ~~left and right hand sides~~ (M, N) and $(M, N-M)$ have same ~~divisors~~ set of common divisors, so have same gcd.

So $gcd(M', N') = gcd(M, N-M)$

Also $N - M > 0$

$\therefore$ $\left(M \leq N \text{ and } A[N-M/M] \right)$ holds in this case

Case 2. Similarly $\left(M > N \text{ and } A[M-N/N] \right)$ holds ✓

Proof of (5) ~~Show~~ $\left(B \wedge M \leq N \right)$ ~~holds~~ implies

~~then~~ $A[N-M/N]$ ~~holds~~.

~~$M \wedge N-M > 0$ and $gcd(M', N') = gcd(M, N-M)$~~

But $\{A[a/X]\}\ X := a\ \{A\}$

is always true:

Substitution Lemma $\sigma \models A[a/X]$ iff $\underbrace{\sigma[\mathcal{A}[\![a]\!]\sigma / X]}_{\mathcal{C}[\![X:=a]\!]\sigma} \models A$

So $\{A[N-M/N]\}\ N := N-M\ \{A\}$ holds

$\therefore$ (5) ✓ Likewise (6) ✓

$D ::= Y := 0;\ (\text{while } X \geq 2 \text{ do } X := X-1;\ X := X-1;\ Y := Y+1)$

$\{X = X' \text{ and } X \geq 0\}\ D\ \{Y = \lfloor X'/2 \rfloor \text{ and } X = \text{rem}(X', 2)\}$

magic A loop-invariant: $X' = 2Y + X$

Hoare logic:

$$\{A\}\ \mathrm{skip}\ \{A\}$$

$$\{A[a/X]\}\ X := a\ \{A\}$$

$$\frac{\{A\}\ c_1\ \{B\}, \quad \{B\}\ c_2\ \{C\}}{\{A\}\ c_1; c_2\ \{C\}}$$

$$\frac{\{A \wedge b\}\ c_1\ \{B\}, \quad \{A \wedge \neg b\}\ c_2\ \{B\}}{\{A\}\ \text{if } b \text{ then } c_1 \text{ else } c_2\ \{B\}}$$

$$\frac{\{A \wedge b\}\ c\ \{A\}}{\{A\}\ \text{while } b \text{ do } c\ \{A \wedge \neg b\}}$$

$$\frac{A \text{ implies } A', \quad B' \text{ implies } B, \quad \{A'\}\ c\ \{B'\}}{\{A\}\ c\ \{B\}}$$

```
EXP ::=  Z:=Y; while X>1 do
           if even(x)
             then (Z := Z×Z ; X := X/2)
             else (Z:=Z×Y; X:=X-1)
```

$A \equiv n = Z^{X}$

$\{X \geq 1 \text{ and } X' = X \text{ and } Y' = Y\}\ EXP\ \{Y'^{X'} = Z\}$

$c \equiv \text{while } b \text{ do } c_0$

Suppose $\{A\}\, c\, \{B\}$ (1)
is valid.

So $A \Rightarrow W(c, B)$ is valid

and $\{W(c,B)\}\, c\, \{B\}$ (2)
is valid.

so by rule of consequence, $\vdash (1)$ providing $\vdash (2)$.

Now let B' be $W(c, B)$. Note that

if $\mathcal{C}[\![c]\!]\sigma \models B'$ ($\sigma \models W(c, B)$)

and $\sigma \models b$

then $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}[\![c]\!](\mathcal{C}[\![c_0]\!]\sigma)$

so $\mathcal{C}[\![c_0]\!]\sigma \models B'$

Note that

$\{W(c, B) \wedge b\}\, c_0\, \{W(c, B)\}$ (3)

is valid,

so by induction $\vdash (3)$

by while rule

$\vdash \{W(c,b)\}\, c\, \{W(c,B) \wedge \neg b\}$

But also $(W(c,B) \wedge \neg b) \Rightarrow B$ (4)

is valid, so $\models (4)$ and $\vdash (3)$ give by rule of consequence yield $\vdash (2)$. ☒

6.044 11/13

~~Simulation~~ Universal Simulator:

~~First idea~~
~~Intuitive idea: Give an IMP Com and a state as input~~

Stored program computer: Want Sim ∈ Com such that when an IMP command configuration $\langle c, \sigma \rangle$ is "load into the store" ~~stored in memory~~ Sim "acts like" c on σ.

~~Two problems: putting c in memory means~~.

~~More precisely:~~ Define $code_{Com}$: ~~Com → Num~~

~~Not possible because Sim can only affect some fixed number of locations.~~

$decode_{Com}$: ~~Com → Num~~ Num → Com $\quad decode(n) = Com_n$

~~$code_{\Sigma} : \Sigma_f \to Num$~~ ~~only finitely many nonzero locations~~

~~$\mathcal{C}[\![Sim]\!]\sigma[mkpair(code(c), code(\sigma))/X_1] = \sigma[$~~

~~$\mathcal{C}[\![Sim]\!]\sigma[n/X_1][code/X_2] = \ldots$~~

$\varsigma(X) = 0$ all $X \in loc$ $\qquad s(n) = \varsigma[n/X_1]$.

$$\mathcal{C}[\![Sim]\!]\, s(mkpair(n_1, n_2)) = s\left(\mathcal{C}[\![Com_{n_1}]\!](s(n_2))X\right)$$

~~$\mathcal{C}[\![Sim]\!]\varsigma[n_1/X_1][n_2/X_2] = \ldots(\mathcal{C}[\![Com_{n_1}]\!](s(n_2))(X_1))$~~

4/20/91

Diverges ∈ Assn

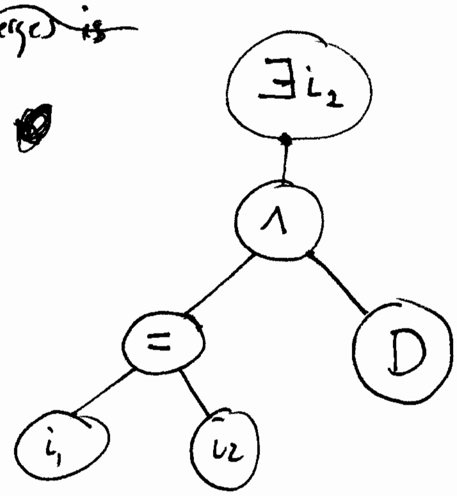
Diverges means "Com_{i_1} diverges in state $s[i_1]$"

• w/log

Diverges of the form $\exists i_2 (i_1 = i_2 \wedge D)$

no i's

~~Gödel # of Diverges is~~



Gödel # of Diverges is

$$mkexists(2, mkconj(mkeq((1,2), \#D)))$$

$1 \to n$

$mkexists(n, j)$
$\equiv mkpair(11, n, j)$

$$mkexist(2, mkconj(mkeq(n,2), \#D)))$$

$=$ Gödel # of Diverges $[n/i_1]$

Exercise 0. Euclid on p 32
Consider $\sigma(M) = 6$ $\sigma(N) = 10$
figure out what each evaluates to.

6.044 F91
L2
9/13/91
p1.

Rules For <u>Aexp</u> & <u>Bexp</u> on Board.

Room change 34-302.

$$\frac{}{\langle n, \sigma \rangle \to n}$$ (crossed out)

Brownie Points For Typos & Feedback about Book.
Reminder about E mail-addresses.

Reading ch's: 1-3.

of people?

Definition By cases. "Evaluates to" relation on Aexp's.

<u>Def</u>: a <u>state</u> σ, σ, is a (total) function from Loc to Num.
σ: Loc $\to$ Num (Σ
Σ = Loc $\to$ Num - set of States.

2+3
$\langle X \pm 3, \sigma \rangle \to \sigma(X) + 3$

triples (a, σ_1, n)
written $\langle a, \sigma \rangle \to n$

$\sigma(M) = 6$ $\sigma(N) = 10$
So
$\langle m, \sigma \rangle \to 6$

~~$\langle (M+N) \times N, \sigma \rangle$~~

derivation tree
establishes that a certain evaluation relation holds.

$$\cfrac{\cfrac{\cfrac{}{\langle M, \sigma \rangle \to 6} \quad \cfrac{}{\langle N, \sigma \rangle \to 10}}{\langle (M+N), \sigma \rangle \to 16} \quad \cfrac{}{\langle N, \sigma \rangle \to 10}}{\langle (M+N) \times N, \sigma \rangle \to 160}$$

~~todo~~ leaves a lot of things unspecified

Com.
$$\frac{\langle a, \sigma\rangle \to n}{\langle X := a, \sigma\rangle \to \sigma'} \quad \text{where } \sigma'(z) = \begin{cases} \sigma(z) & \text{o/w} \\ n & \text{if } z = x \end{cases}$$

$\sigma[n/x]$

evals to relation for Com of type Com, state, state

notation of function patching.

$\sigma' =_{df} \sigma[n/x]$

$$\frac{}{\langle skip, \sigma\rangle \to \sigma}$$

~~$\langle$ if b then c_0 else $c_1, \sigma\rangle \to \sigma'$~~

$$\frac{\langle b, \sigma\rangle \to true, \quad \langle c_0, \sigma\rangle \to \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \to \sigma'}$$

$\langle b, \sigma\rangle \to$ false similarly.

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

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

6.044 L3
p1
9/16/91

SHEUNG LI – Possible Jury Duty on Wed.
22 people

Euclid ::=
while ¬(M=N) *do* c

---

c ::= *if* M ≤ N
then N := N−M
else M := M−N

(conditional evaluation)

$$\frac{\overbrace{\langle b,\sigma\rangle}^{\text{configuration}} \to true, \quad \overbrace{\langle c_0,\sigma\rangle}^{\text{configuration}} \to \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \to \sigma'}$$

(while^T) ✓

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

(while^F)

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

Today make sure we understand both operational and intuitive notion of → (evals to relation)

looking for inductive definitions.

---

administrative review of diagnostic exam:

F: A → B injection: $f(x) = f(y) \Rightarrow x = y$

care with distributing textbook.

email-communication

---

above
Configuration: pair of ~~program~~ code + state.

i

"Implicit algorithmic search procedure to find derivation tree"

suppose σ is st

$\sigma(M) = 6, \quad \sigma(N) = 10$

Search for a derivation tree that would enable us to deduce what Euclid evaluates to.

⑩ $\langle N,\sigma\rangle \to 10 \; \langle M,\sigma\rangle \to 6$

③ $\langle M,\sigma\rangle \to 6 \;\; \langle N,\sigma\rangle \to 10$ ⑧ $\langle M,\sigma\rangle \to 6 \; \langle N,\sigma\rangle \to 10$ $\langle N-M,\sigma\rangle \to 4$

② $\langle (M=N),\sigma\rangle \to false$ ④ ⑦ $\langle (M \le N),\sigma\rangle \to t$ ⑨ $\langle N := N-M,\sigma\rangle \to \sigma[4/N]$

① $\langle \neg(M=N),\sigma\rangle \to true$ ⑤, ⑥ $\langle c,\sigma\rangle \to \sigma[4/N]$, ⑥ $\langle Euclid, \sigma[4/N]\rangle \to$

$\langle Euclid, \sigma\rangle \to \sigma[2/M][2/N]$

) $\langle 4$ out & put below

① Don't know which is right rule for while to use, but evaluating $\langle b,\sigma\rangle$ will let us know.

after step ⑤ we now know to use rule (while-t)

Q: are inference rules read from left to right?
A: NO.
Derivation tree exists or doesn't abstractly. given one we can check locally that it is right rules don't tell us strategy.

Exercise. O Exercise O. due on Friday

$\langle Euclid, \sigma[4/N]\rangle \to$

6.044 L3
p2
9/16/91

It was not clear off hand from form of rules there was a nice simple search algorithm

Very abstract leaving a lot of details to programmer.

Should be obvious to check that a derivation is correct linear time to verify derivation tree.

These rules: if there is a derivation tree we can find it in time proportional to size of tree.

** Derivations can be found algorithmically in "time" proportional to their "size".

Derivations can obviously be checked in linear time.

example

$$\frac{\langle b, \sigma\rangle \to \text{False}}{\langle b \wedge b', \sigma\rangle \to \text{false}}$$

& could easily be a rule but isn't.
we could add a rule like this

derivation tree is still checkable.
but if we added both above rule and

$$\frac{\langle b', \sigma\rangle \to \text{False}}{\langle b \wedge b', \sigma\rangle \to \text{false}}$$

adding these rules to system gives you a new relation $\to$
you no longer have nice way of finding derivations.

rules as we wrote them have this special character.

(about our rules) – not for crossed out ones. $\langle b, \sigma \rangle$

Thm: For every (boolean) arithmetic configuration $\langle a, \sigma \rangle$ there is a unique n / t such that $\langle a, \sigma \rangle \to n \in$ integer, $\langle b, \sigma \rangle \to t \in$ truthvalue

again, Not true in general.

eg
$$\frac{\langle a_0, \sigma \rangle \to n}{\langle choose(a_0, a_1), \sigma \rangle \to n} \qquad \frac{\langle a_1, \sigma \rangle \to n}{\langle choose(a_0, a_1) \rangle \to n}$$

way in code to indicate choice of implementation to some degree.

~~Thm:~~ Imp is deterministic

Thm: For every command configuration $\langle c, \sigma \rangle$, there is at most one σ' s.t. $\langle c, \sigma \rangle \to \sigma'$.

Lemma For all σ, $c \in$ Com there is no σ' s.t. $\langle$ while true do $c, \sigma \rangle \to \sigma'$

pf: by contradiction, Choose c, σ with shortest derivation of $\langle w, \sigma \rangle \to \sigma'$ for some σ' where $w \equiv$ while true do c.

we must have

$$\frac{\langle true, \sigma \rangle \to true, \quad \langle c, \sigma \rangle \to \sigma'', \quad \langle w, \sigma'' \rangle \to \sigma'}{\langle w, \sigma \rangle \to \sigma'}$$

(a smaller derivation!)

In order for this derivation to exist there must be a derivation of $\langle w, \sigma'' \rangle \to \sigma'$

Minimum Principle $\equiv$ Induction Principle.

1/18/91 Notes For 6.044 Lecture 4.
Write Comm rules on Left Board.

Over the past 3 lectures Albert has given you the syntax of IMP and defined ~~to~~ ~~$\to$~~ $\to$ to operationally tell you how IMP code behaves.

Well today I'm going to give you an alternate definition of how IMP code behaves.

I'll define a relation $\to_1$ – rewrites in 1 step – which tries to capture single steps in evaluation of code.

~~what is a single step.~~

~~Here~~ ~~by~~ now I'll have rules of the form

$\langle a, \sigma\rangle \to_1 \langle a', \sigma'\rangle$

and ~~$\langle a, \sigma\rangle \to_1 n$~~.

Sampling of rules for Aexps:

$$\langle X, \sigma\rangle \to_1 \langle \sigma(X), \sigma\rangle \quad *$$

$$\frac{\langle a_0, \sigma\rangle \to_1 \langle a_0', \sigma\rangle}{\langle a_0 + a_1, \sigma\rangle \to_1 \langle a_0' + a_1, \sigma\rangle}$$

$$\frac{\langle a_1, \sigma\rangle \to_1 \langle a_1', \sigma\rangle}{\langle n + a_1, \sigma\rangle \to_1 \langle n + a_1', \sigma\rangle}$$

$$\langle n + m, \sigma\rangle \to_1 \langle p, \sigma\rangle *$$

Similar rules for $-$, $\times$

~~for convenience we'll also have a rule of the form $\langle n, \sigma\rangle \to_1 n$ just to make it clear we're done evaluating.~~

we could make the same sorts of rules for B exps.

e.g.

$$\frac{\langle b,\sigma\rangle \to_1 \langle b',\sigma\rangle}{\langle \neg b,\sigma\rangle \to_1 \langle \neg b',\sigma\rangle}$$

$$\langle \neg t,\sigma\rangle \to_1 \langle f,\sigma\rangle$$

$$\langle \neg f,\sigma\rangle \to_1 \langle t,\sigma\rangle$$

~~How~~ problem 1 on your problem set assignment was to do this for commands.

Similarly for commands you have steps of the form

$$\langle c,\sigma\rangle \to_1 \langle c',\sigma'\rangle$$

also can have "special" steps of form

$\langle c,\sigma\rangle \to \sigma'$ when after doing c there is nothing left to do,

~~eg~~ viz. c consists of only a single instruction

just to get a flavor of what it looks like:

$$\langle X:=5; Y:=1, \sigma\rangle \to_1 \langle Y:=1, \sigma[5/x]\rangle \to_1 \sigma[5/x][1/y]$$

there are some ambiguities as what we should take to be one step.

For example if

$\langle b,\sigma\rangle \to_1 \langle true,\sigma\rangle$ ← different from text

should we have

$$\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma\rangle \to_1 \langle c_1,\sigma\rangle$$

or

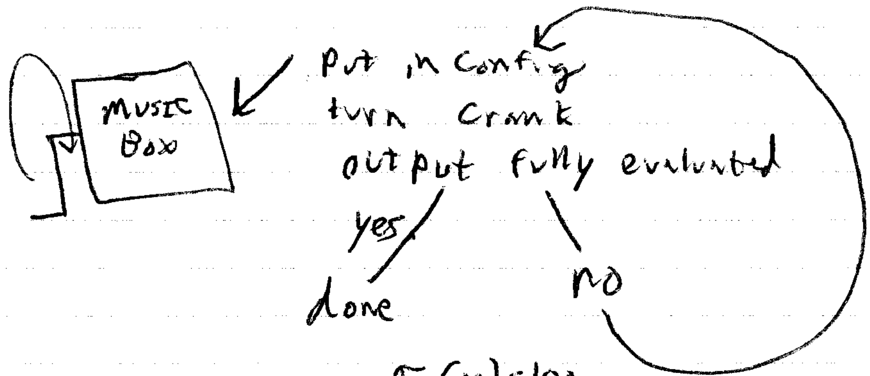
(b ≢ true) $\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma\rangle \to_1$ ~~$\langle c_1,\sigma\rangle$~~ $\langle \text{if } true \text{ then } c_1 \text{ else } c_2,\sigma\rangle$?

For IMP it does not matter...

But in, for example, parallel languages, it does...

We have said how to execute ~~1 step of~~ code by 1 step.

we have a crank, plug in con.



$\sigma(x) = 100$

eg $\langle (5+6) \times 4 + x, \sigma \rangle \to_1 \langle 11 \times 4 + x, \sigma \rangle \to_1 \langle 44 + x, \sigma \rangle$

$\to_1 \langle 44 + 100, \sigma \rangle \to_1 144.$

~~This~~ It is only natural to define reln $\to_1^*$
lets you turn crank 0 times
or 1 time
or 2 times
⋮

$\langle a, \sigma \rangle \to_1^* \langle a', \sigma' \rangle$ if there is some seq of $\to_1$ steps that gets you there.

It is really easy to see how to implement this ~~via a recursive procedure~~ in scheme or say lisp.
The time to find ~~a~~ what $\langle c, \sigma \rangle \to_1$ if anything is limited to the length of c (sort of)

Not covered → where in general we can't tell w/ $\to$.

Postpone? { although $\to_1$ might not stop.
eg $\langle$while true do skip, $\sigma\rangle \to_1 \langle$if true do skip; while true do skip, $\sigma\rangle$
$\to_1$ $\langle$skip; while true do skip, $\sigma\rangle$ $\to_1$ (back to start)

haven't gotten rid of problems, just pushed them somewhere else

I would like to say that lang defined by $\to_1^*$ is "the same" as that defined by $\to$.

This captured by following fact

Fact
$\langle a, \sigma\rangle \to_1^* \langle n, \sigma\rangle$ iff $\langle a, \sigma\rangle \to n$ A exps
$\langle b, \sigma\rangle \to_1^* \langle v, \sigma\rangle$ iff $\langle b, \sigma\rangle \to v$ B exps
$\langle c, \sigma\rangle \to_1^* \sigma'$ iff $\langle c, \sigma\rangle \to_1^* \sigma'$ C exps.

An important step in proving this is to show that
$\langle w, \sigma\rangle \to \langle \text{if } b \text{ then } (c; w) \text{ else skip}, \sigma\rangle$ is ok.
Discussion of why this rule was necessary
intuitively this is ok because w & if b then c;w "behave the same".

We define $\sim$: command equivalence - which captures exactly what it is about them that is the same.

specifically in general
$c_1 \sim c_2$ iff $\forall \sigma, \sigma' \in \Sigma$, $\langle c_1, \sigma\rangle \to \sigma'$ iff $\langle c_2, \sigma\rangle \to \sigma'$.

So we prove the following Lemma:

Lemma: Let $w \equiv \text{while } b \text{ do } c$. $w \sim \text{if } b \text{ then } (c; w) \text{ else skip}$

pf: Let σ, σ' be arbitrary states.
Show $\langle w, \sigma\rangle \to \sigma'$ iff $\langle \text{if } b \text{ then } (c; w) \text{ else skip}, \sigma\rangle \to \sigma'$.

$\Rightarrow$

Suppose $\langle w, \sigma\rangle \to \sigma'$. End of deriv must look like either

Big Idea!! Transform derivation of $\langle w, \sigma\rangle \to \sigma'$ into derivation of $\langle \text{if } b \text{ then } (c; w) \text{ else skip}, \sigma\rangle \to \sigma'$

$$\frac{\langle b, \sigma\rangle \to \text{false}}{\langle w, \sigma\rangle \to \sigma} \quad \text{or} \quad \frac{\langle b, \sigma\rangle \to \text{true}, \quad \langle c, \sigma\rangle \to \sigma'', \quad \langle w, \sigma''\rangle \to \sigma'}{\langle w, \sigma\rangle \to \sigma'}$$

(1) and so $\sigma = \sigma'$ (σ' must be σ). ← not too hard

(2)

(big idea) transform legal derivation of
$\langle w, \sigma \rangle \rightarrow \sigma'$ into legal d.

$$\frac{\boxed{A} \quad \boxed{B} \quad \boxed{C}}{\frac{}{\langle b,\sigma\rangle \to true}, \quad \langle c,\sigma\rangle \to \sigma'' \quad \langle w,\sigma''\rangle \to \sigma'}$$

$$\frac{\langle b,\sigma\rangle \to true, \quad \langle c,\sigma\rangle \to \sigma'' \quad \langle w,\sigma''\rangle \to \sigma'}{\langle w,\sigma\rangle \to \sigma'}$$

build derv of

$$\frac{\dfrac{\boxed{A}}{\langle b,\sigma\rangle \to true}, \quad \dfrac{\dfrac{\boxed{B}}{\langle c,\sigma\rangle \to \sigma''}, \quad \dfrac{\boxed{C}}{\langle w,\sigma''\rangle \to \sigma'}}{\langle c;w,\sigma\rangle \to \sigma'}}{\langle \text{if } b \text{ then } c;w \text{ else skip}, \sigma\rangle \to \sigma'.}$$

Case 1 similarly. (but easier)

not covered

$\Leftarrow$ Show $\langle$if b then $c;w$ else skip, $\sigma\rangle \to \sigma'$ implies $\langle w,\sigma\rangle \to \sigma'$.

Transform Again!!
what does deriv of $\langle$if..., $\sigma\rangle \to \sigma'$ look like?

$$\frac{\langle b,\sigma\rangle \to false, \quad \langle skip,\sigma\rangle \to \sigma}{\langle \text{if } b \text{ then } c;w \text{ else skip},\sigma\rangle \to \sigma}$$

(1) and so $\sigma = \sigma'$
as all skip conditions go to σ.

or

$$\frac{\dfrac{\boxed{D}}{\langle b,\sigma\rangle \to true} \quad \dfrac{\boxed{E}}{\langle c;w,\sigma\rangle \to \sigma'}}{\langle \text{if } b \text{ then } c;w \text{ else skip},\sigma\rangle \to \sigma'}$$

construct derivation of

not covered

don't have derive of here yet

we must further subdivide derivation E.

$$\dfrac{\dfrac{D}{\langle b,\sigma\rangle \to true} \quad \dfrac{F}{\langle c,\sigma\rangle \to \sigma''} \quad \dfrac{G}{\langle w,\sigma''\rangle \to \sigma'}}{\langle w,\sigma\rangle \to \sigma'}$$

do

What must E look like.
well

$$\dfrac{\dfrac{F}{\langle c,\sigma\rangle \to \sigma''} \quad \dfrac{G}{\langle w,\sigma''\rangle \to \sigma''}}{\langle c;w,\sigma\rangle \to \sigma'}$$

---

if Time do case 1.

$$\dfrac{\dfrac{H}{\langle b,\sigma\rangle \to false} \quad \langle skip,\sigma\rangle \to \sigma}{\langle \text{if } b \text{ then } c;w \text{ else } skip,\sigma\rangle \to \sigma}$$

$$\dfrac{\dfrac{H}{\langle b,\sigma\rangle \to false}}{\langle w,\sigma\rangle \to \sigma'}$$

use remaining time for Banter / Q & A / simple confusion.

6.044L5
p1 Discussion about Quiz Schedule.
9/20/91

Completion of $w \sim$ if b then c;w else skip.

Talk about methodology of communicating results.
What a rigorous checkable proof,
May not be understandable, reproducible.

"Hidden curriculum".

Structural induction

Top level: just 1 kind of induction - induction of a definition of a set.

Def 0 is a non-negative integer, and
if n is, so is n+1.

From no assumptions 0 is ..
from assuming n is ..

Induction principle
If P() is a property of non negative integers,
such that P(0) and for every non negative integer n
if P(n) then P(n+1)
then P holds for all nonnegative integers.

This principle generalizes.

Generally, if a set S defined inductively by "rules",
and a property of elements in the set is preserved
by all the rules then every point in the
set has the property.

Define R^* reflexive transitive closure of a relation $R \subset D \times D$.

"$d_1 R d_2$" $\leftrightarrow$ $(d_1, d_2) \in R$.

Handout 9 — $\to_1$ Rules for Com

11 — Proof of following Thm.

6.044L6
9/13
p1

Theorem: $\forall a, n, \sigma \; \langle a,\sigma\rangle \to_1^* \langle n,\sigma\rangle$ iff $\langle a,\sigma\rangle \to n$.

Lemma: $\langle a,\sigma\rangle \to_1 \langle a',\sigma\rangle \;\&\; \langle a',\sigma\rangle \to n \Rightarrow \langle a,\sigma\rangle \to n$

pf by structural induction
show it holds for
4 norm

1) $a \equiv m$
2) $a \equiv X$
3) $a \equiv a_0 + a_1 \quad a_0 \not\equiv m$
4) $a \equiv n_0 + a_1$

easy case $a \equiv m$.
1) $\langle a,\sigma\rangle \not\to_1$ so lemma holds vacuously.

3) $a \equiv a_0 + a_1 \quad a_0 \notin Num$
assume lemma holds for a_0, a_1
so if $\langle a_0,\sigma\rangle \to_1 \langle a_0',\sigma\rangle \;\&\; \langle a_0',\sigma\rangle \to n_0 \Rightarrow \langle a_0,\sigma\rangle \to n_0$

$\exists\, a_0'$ st $\langle a_0 + a_1,\sigma\rangle \to_1 \langle a_0' + a_1,\sigma\rangle$
& $\langle a_0,\sigma\rangle \to_1 \langle a_0',\sigma\rangle$.

$\exists n_0, n_1$.

well $\dfrac{\langle a_0',\sigma\rangle \to n_0 \quad \langle a_1,\sigma\rangle \to n_1}{\langle a',\sigma\rangle \to n}$ $\quad n_0 + n_1 = n$.

so we have $\langle a_0,\sigma\rangle \to_1 \langle a_0',\sigma\rangle$
& $\langle a_0',\sigma\rangle \to n_0$
so by ind'n $\langle a_0,\sigma\rangle \to n_0$

so $\dfrac{\langle a_0,\sigma\rangle \to n_0 \quad \langle a_1,\sigma\rangle \to n_1}{\langle a_0 + a_1,\sigma\rangle \to n.}$ $\quad n = n_0 + n_1$.

Showing $(\Rightarrow)$ of theorem:

$\langle a, \sigma\rangle \to_1^* \langle n, \sigma\rangle$

$\langle a, \sigma\rangle \to_1 \langle a_1, \sigma\rangle \to_1 \cdots \to_1 \langle n, \sigma\rangle$ (with m below the chain)

Chain is just $\langle n, \sigma\rangle$ $\langle n, \sigma\rangle$

Base: $\langle n, \sigma\rangle \to n$ ☒

$\langle a', \sigma\rangle \to_1^{m-1} \langle n, \sigma\rangle \Rightarrow \langle a', \sigma\rangle \to n$

formin applying induction (i.e. terminology w/ quantifiers) nicer to use indn on defn of $\to_1^*$

1 vote for Tuesday

$\Leftarrow$:

pf by indn on structure of a.

$a \equiv a_0 + a_1$ by defn of $\to$ $\langle a, \sigma\rangle$

$$\frac{\langle a_0, \sigma\rangle \to n_0 \quad \langle a_1, \sigma\rangle \to n_1}{\langle a_0 + a_1, \sigma\rangle \to n} \qquad n_0 + n_1$$

$\langle a_0, \sigma\rangle \to_1^* \langle n_0, \sigma\rangle$
$\langle a_1, \sigma\rangle \to_1^* \langle n_1, \sigma\rangle$ } by indn

Remark
$$\frac{\langle a_0, \sigma\rangle \to_1 \langle a_0', \sigma\rangle}{\langle a_0 + a_1, \sigma\rangle \to_1 \langle a_0' + a_1, \sigma\rangle} \qquad \frac{\langle a_0, \sigma\rangle \to_1^* \langle n_0, \sigma\rangle}{\langle a_0 + a_1, \sigma\rangle \to_1^* \langle n_0 + a_1, \sigma\rangle}$$

By induction on defn of $\to_1^*$ we can show similarly
$$\frac{\langle a_1, \sigma\rangle \to_1^* \langle n_1, \sigma\rangle}{\langle n_0 + a_1, \sigma\rangle \to_1^* \langle n_0 + n_1, \sigma\rangle}$$

$R^* \subset D \times D$

inductive def: $d \, R^* \, d$ for all d,

and if $d_1 \, R \, d_2$ and $d_2 \, R^* \, d_3$ then $d_1 \, R^* \, d_3$

"how does d_1 get to d_3"

R^* you are related by a chain of some finite # of R steps, possibly none.

$d_1 \, R^* \, d_2$ iff $d_1 = d_2$ or there are elts $d^0, d^1, d^2, \ldots, d^n$

s.t. $d_1 = d^0 \; R \, d^1 \, R \, d^2 \; R \cdots R \, d^n = d_2$

Structural Induction: The induction principle applied to a set of terms defined (inductively) by a grammar

$a ::= n \mid X \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2$

Proposition: $\langle a, \sigma \rangle \to n$ iff $\langle a, \sigma \rangle \to_1^* \langle n, \sigma \rangle$

pf: ($\Leftarrow$)

> Lemma: if $\langle a, \sigma \rangle \to_1 \langle a', \sigma \rangle$ and $\langle a', \sigma \rangle \to n$, then $\langle a, \sigma \rangle \to n$
>
> Pf: Structural induction on a

Using the Lemma, by trivial proof by induction on the definition of $(\to_1)^*$, we conclude ($\Leftarrow$).

Take up on Monday

$\Rightarrow$ S. on def of evaluates to.

6.044L6
9/23/91
p2.

suppose $\langle a_0, \sigma \rangle \rightarrow_1 \langle a_0', \sigma \rangle \rightarrow_1 \langle a_0'', \sigma \rangle \cdots \rightarrow_1 \langle n_0, \sigma \rangle$

$\overline{\langle a_0 + a_1, \sigma \rangle \rightarrow}$

$\langle a_0 + a_1, \sigma \rangle \rightarrow_1^* \langle n_0 + a_1, \sigma \rangle \rightarrow_1^* \langle n_0 + n_1, \sigma \rangle \rightarrow_1 \langle n, \sigma \rangle$

$\langle a_0 + a_1, \sigma \rangle \rightarrow_1^* \langle n, \sigma \rangle$.

Quiz. Thursday 7-9 pm.

6.044 L7 (Wald)
p1
9/25/91

Addendum to PS 2.

You may assume: ...

$\forall a \in (Aexp - Num)\ \forall \sigma \in \Sigma\ \exists!\, a'$ st $\langle a, \sigma \rangle \to \langle a', \sigma \rangle$

$\forall b \in Bexp - \{true, false\}\ \forall \sigma \in \Sigma\ \exists!\, b'$ st $\langle b, \sigma \rangle \to \langle b', \sigma \rangle$

Induction on Derivations:

Ind on Nat $\Rightarrow P(0)$ $\quad \forall n\ P(n-1) \Rightarrow P(n)$

Structural ind'n $\Rightarrow P(n)$ $\quad P(X)$ $\quad \forall (a_0 + a_1):\ P(a_0)$ and $P(a_1) \Rightarrow P(a_0 + a_1)$

Today: Induction on Derivations

$\forall c, \sigma_0, \sigma, \sigma_1.$ If $\langle c, \sigma_0 \rangle \to \sigma$
and $\langle c, \sigma_0 \rangle \to \sigma_1$ then $\sigma = \sigma_1$

pf by ind'n on deriv'n of $\langle c, \sigma_0 \rangle \to \sigma$

Cases: forms of deriv'n of $\langle c, \sigma_0 \rangle \to \sigma$

$$d = \frac{\vdots \quad \vdots \quad \vdots}{\langle c, \sigma_0 \rangle \to \sigma}$$

$$d_1 = \frac{\vdots \quad \vdots \quad \vdots}{\langle c, \sigma_0 \rangle \to \sigma_1}$$

$c \equiv skip$
$c \equiv X := a$
$c \equiv c_0 ; c_1$
$c \equiv$ if b then c_0 else
$c \equiv$ while b do c_0

$$d \equiv \frac{}{\langle skip, \sigma_0 \rangle \to \sigma_0}$$

so $\sigma = \sigma_0 = \sigma_1$

$d_1 \equiv \langle skip, \sigma_0 \rangle \to \sigma_0$

Case of $X := a$ similarly.

case of while loop.

appeal to similar result for Aexp, Bexp

so $\exists! v$ st $\langle b, \sigma\rangle \to v$

$$d = \frac{\langle b, \sigma_0\rangle \to \text{false}}{\langle \text{while } b \text{ do } c, \sigma_0\rangle \to \sigma_0}$$

$$d_1 = \frac{\langle b, \sigma_0\rangle \to \text{false}}{\langle \text{while } b \text{ do } c, \sigma_0\rangle \to \sigma_1 = \sigma_0}$$

so $\sigma = \sigma_0 = \sigma_1$

hard case

$$d = \frac{\dfrac{\vdots}{\langle b, \sigma_0\rangle \to \text{true}} \quad \dfrac{\vdots}{\langle c, \sigma_0\rangle \to \sigma'} \quad \dfrac{\vdots}{\langle \text{while } b \text{ do } c, \sigma'\rangle \to \sigma}}{\langle \text{while } b \text{ do } c, \sigma_0\rangle \to \sigma}$$

$$d' = \frac{\dfrac{\vdots}{\langle b, \sigma_0\rangle \to \text{true}} \quad \dfrac{\vdots}{\langle c, \sigma_0\rangle \to \sigma_1' = \sigma'} \quad \dfrac{\vdots}{\langle \text{while } b \text{ do } c, \sigma_1' = \sigma'\rangle \to \sigma_1 = \sigma}}{\langle \text{while } b \text{ do } c, \sigma_0\rangle \to \sigma_1}$$

erasing is bad in teaching an undergraduate class.

Section 3.4 in excruciating detail.

discussion about circularity

- appeal to 6.001 intuitions.

6.044L6
p2
9/25/91

Rule Induction:

$$\frac{}{0} \qquad \frac{n}{n+1}$$

Basic Principle: if you have a set defined by some set of rules:

$$\frac{x_1 \quad x_2 \quad x_3 \dots}{y}$$

$$[\forall x_i \; P(x_i)] \Rightarrow P(y)$$

must show $P(0)$

$$P(n) \Rightarrow P(n+1)$$

any → triple has a unique derivation
For any rule system w/ such a property,
rule induction is roughly "the same" as induction on derivations.

Quiz Question

Course of values more:
give us Strong induction as a Rule induction.

$$\frac{}{\{0\}} \qquad \frac{S}{S \cup \{max(S)+1\}}$$

6.044L9

p1 9/30

Def. $loc_L : Com \to Pow(Loc)$

By induction on def. of Com: — Beg of Sect 3.5 in text page 37.

$loc_L(skip) ::= \emptyset$

$loc_L(X := a) ::= \{X\}$

$loc_L(c_0 ; c_1) ::= loc_L(c_0) \cup loc_L(c_1)$

$loc_L(\text{if } b \text{ then } c_0 \text{ else } c_1) ::=$ "

$loc_L(\text{while } b \text{ do } c) ::= loc_L(c)$

~~Do~~ Today's topic: Inductive def's.

Def. of <u>functions</u> by induction.

Reminder about how to define sets inductively.

"the smallest set you can get by only putting in just what you have to".

Concrete ~~set~~ example inductive defn of non-neg integers.

Put in 0.
if n is in there put in n+1.
non neg ints smallest set closed under this,
but real numbers closed under rules.. but not <u>the</u> set defined like this.

defn of a set closed under the rules.

defined smallest set by taking intersection of all sets closed under the rules.

$\bigcap$ {sets closed under R} is closed under R.

implicit 3^{rd} condition in defining a set inductively
or (extremal clause) = "nothing else is in the set except those elements implied by the preceding rules".

What can go wrong defining functions by induction.

eg. example of such a definition of loc.

underlying set in this case com, defined itself by induction.

What is a point of confusion with defining inductive def's

double ::= 0
double (m+1) ::= 2 + double (n)

} an fcn defined by by inducn on underlying set...

Fact: double : $\mathbb{N} \to \mathbb{N}$
double (n) = 2n.

→ double (n) = if n = 0 then 0 else 2 + double(n-1)

$$f(n+m) = \begin{cases} 1 & \text{if } n+m \leq 1, \\ f(n)+f(m) & \text{otherwise} \end{cases}$$

n, m ≥ 0

if ~~n+m~~ > 1 if n > 0 and m > 0
~~and either n or m > 0~~

goes wrong for n = 0 m = 2

k	0	1	2	3	4
f(k)	1	1	2	3	4
			~~1+1~~	1+2	2+2
				2+1	1+3
					3+1

$$f(n+m) = \begin{cases} 1 & \text{if } n+m \leq 1 \\ f(n)+f(m) & \text{if } n > 0 \text{ and } m > 0 \end{cases}$$

$$f(k) = \begin{cases} 1 & \text{if } k = 0 \\ k & \text{ow.} \end{cases}$$

$$g(n+m) = \begin{cases} 1 & \text{if } n = m = 0 \\ 1 & \text{if } n+m \leq 1 \text{ and either } n \text{ or } m > 0 \\ g(n) + 2 * g(m) & n > 0 \text{ and } m > 0 \end{cases}$$

looks as safe & satisfactory as definition of f.

6.044J
p2
9/30/91

k	0	1	2	3
g(k)	1	1	3 $(1+2*1)$	$(1+2*3)$ 7 / $(2+1*3)$ 5

↑ ouch!!

g(2): 1+0 g(3): 1+2, 2+1

What caused the problem:
we were trying to define value of f(k) in terms of how you split up k

def. of a function by induction on an inductively defined set is ok as long as derivations are unique.

was going to prove

Lemma Ch 4. Prop. 8

if $Y \notin loc_L(c)$, and $\langle c, \sigma \rangle \to \sigma'$, then $\sigma(Y) = \sigma'(Y)$.

Forum Mail decides if we do this tomorrow

G(044 L10
10/2/91
p1

Prop. 8. (p. 45) $Y \notin loc_L(c)$ & $\langle c, \sigma \rangle \to \sigma'$ implies $\sigma(Y) = \sigma'(Y)$. (There are no side-effects on locations not on the lefthandside of assignment statements in a command).

We expect Prop 8 to hold. It is a sanity check did we get the definitions right.

Pf: By "rule induction" on def. of evals to : $\to$.

Trick: finding right notion of Red.

inductively defined set is $\{ \langle c, \sigma, \sigma' \rangle \mid \langle c, \sigma \rangle \to \sigma' \}$

↑ a point

a red point; is a point $\langle c, \sigma, \sigma' \rangle$ s.t.: ~~$X \in loc_L(c)$ & $\langle c, \sigma \rangle \to \sigma'$~~

$\nearrow$ $\forall Y \in loc \; [Y \notin Loc_L(c) \;\&\; \langle c, \sigma \rangle \to \sigma' \text{ implies } \sigma(Y) = \sigma'(Y)]$

an assertion whose truth or falsity depends on c, σ, σ'.
ie call this red

Prove by rule induction that all points are red.

Base cases (axioms):
$[\langle c, \sigma, \sigma' \rangle$ is a point because $c \equiv skip$ and $\sigma \equiv \sigma'_0]$
that is axiom: $\langle skip, \sigma \rangle \to \sigma$.
but $\langle skip, \sigma, \sigma' \rangle$ is red since ~~$\sigma(Y)$~~ $\sigma = \sigma'$. So $\sigma(Y) = \sigma'(Y)$.

$[c \equiv X := a$, and $\langle a, \sigma \rangle \to n$ and $\sigma' \equiv \sigma[n/X].]$

$\langle X := a, \sigma, \sigma[n/X] \rangle$ is red because.
(1). If $Y \notin Loc_L(c)$, then $Y \notin \{X\}$ by def of loc_L.
That is $Y \not\equiv X$.
So (2) $\sigma(Y) = \sigma[n/X](Y)$ by def. of $\sigma[n/X]$ ☒

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

induction step

by rule:

[$\langle c, \sigma, \sigma' \rangle$ is a point
$\langle b, \sigma\rangle \to true$ & $\langle c', \sigma, \sigma''\rangle$ is a point & $\langle c, \sigma'', \sigma'\rangle$

$c \equiv$ while b do c'

Now assume by rule induction, that $\langle c', \sigma, \sigma''\rangle$ is red
and $\langle c, \sigma'', \sigma'\rangle$ is red;
and we must show that
$\langle c, \sigma, \sigma'\rangle$ is red.

So suppose $Y \notin loc_L(c)$, then $Y \notin loc_L(c')$ by def. of Loc_L.
so by induction, $\sigma(Y) = \sigma''(Y)$

also, by induction, $\sigma''(Y) = \sigma'(Y)$
$\therefore \sigma(Y) = \sigma'(Y)$ QED ■

Equivalence of all inductions.

suppose a set I_R is defined inductively by a set R of rules (or rule "instances"), where elements of R are ordered pairs of the form $\langle X, y\rangle$ where X is a finite set. Rules are written "X/y"
or $\dfrac{X}{y}$ or $\dfrac{x_1, x_2, \ldots, x_n}{y}$

Def. a derivation, d for R.
(1) $\emptyset/y$ is a derivation of y if $\emptyset/y \in R$
(2) if $d_1, \ldots, d_n$ are derivations of $x_1, \ldots, x_n$
and $\{x_1, \ldots, x_n\}/y \in R$ then $\langle d_1, \ldots, d_n, \{x_1, \ldots, x_n\}/y\rangle$ is a derivation of y.

Fact: $y \in I_R$ iff there exists an R-derivation of y.

P.S. 1 Stats

	# Submitted	High	Low	Median
exercise 0	23	10	7	10
1	22	10	1	2.5
2	21	10	1	4

6.044L10
10/4/11
p1

Discussion of ≡ as Syntactic identity

$\to_1^*$ R^*

$\langle c_0; c_1, \sigma\rangle \to_1^* \sigma'$ ⇒ $\exists \sigma''$ $\langle c_0, \sigma\rangle \to_1^* \sigma''$
and $\langle c_1, \sigma''\rangle \to_1^* \sigma'$.

to prove red($\langle a,b\rangle$) forall $\langle a,b\rangle \in R^*$

show 1) show red $\langle a,a\rangle$ forall relevant a.
and 2) show if $\exists c$ aRc and cR^*b and Red($\langle c,b\rangle$),
then red($\langle a,b\rangle$).

Flow here's a little imp program.

devmore to evals to rewrite to to state are unified.

Discussion of difference between
polynomial as recipe for computing
+ polynomial as a function.

Fixpoints.
$a \in A$ is a __fixed point__ of $f: A \to A$ iff
$f(a) = a$

factorial = ---- factorial ----
Bigider
factorial = f (factorial)

all continuous functions ~~don't~~ have fixed point.

understanding Fixed point!!

Induction & fixed points:

setup: rules R of the form $\{x_1, \ldots, x_n\}/y$

I_R = set defined inductively by R, namely,

$\{y \mid$ there is an R-derivation of $y\}$.

$\Vdash_R y$

$d \Vdash y$ d is a derivation of y.

another way to describe:

def. θ is R-closed:
$\forall n \geq 0\ \forall$ rules $\{x_1, \ldots, x_n\}/y \in R$, $x_i \in \theta$ for $i = 1, \ldots, n$ implies $y \in \theta$.

special case $n=0$ axiom.

Lemma: I_R = the least set which is R-closed,

(implicit assum - there is a smallest set).

Namely $I_R = \bigcap \{\theta \mid \theta$ is R-closed$\}$.

Talk through it. what is to be proven?

Step 1. Show there is some set which is R-closed. (take for granted).
Step 2. Show if I take all of the R-closed sets their intersection is an R-closed set.

6.044J/0
10/4/91 ~~Lemma:~~ ← a functn.
p 2

Def: $\hat{R}$: Sets → Set, where $\hat{R}(A) =_{def} \{y \mid \forall \{x_1, \ldots, x_n / y\} \in R.\ x_i \in A \text{ for } i = 1, \ldots, n\}$

Lemma: B is R-closed iff $\hat{R}(B) \subseteq B$.

pf. by def. (of R-closed).

Lemma: $\hat{R}$ is monotone. (preserve order). $A \subset A' \Rightarrow \hat{R}(A) \subset \hat{R}(A')$

$\emptyset \subset \hat{R}(\emptyset)$

$\hat{R}(\emptyset) \subset \hat{R}(\hat{R}(\emptyset))$

$\hat{R}(\hat{R}(\emptyset)) \subset \hat{R}^{(3)}(\emptyset)$

Lemma $\bigcup_n \hat{R}^{(n)}(\emptyset)$ is the least R-closed set.

6.044L12
10/9/91
p1

Denotational Semantics.

$\left\{\begin{matrix} b \\ a \\ \langle c,\sigma\rangle \end{matrix}\right. \to \begin{matrix} t \\ n \\ \sigma' \end{matrix}$ iff $\left\{\begin{matrix} b \\ a \\ \langle c,\sigma\rangle \end{matrix}\right. \to_1^* \begin{matrix} \langle t,\sigma\rangle \\ \langle n,\sigma\rangle \\ \sigma' \end{matrix}$

Talk about Implementability. Why these are "operational semantics".
recipies for pushing symbols around.

Motivation for denotational semantics.
a thing to associate w/ command fcn from σ to σ.

$\mathcal{C}[\![c]\!]$ instead of meaning(c).

$$\mathcal{C}[\![c]\!] = \{ (\sigma, \sigma') \mid \langle c,\sigma\rangle \to \sigma' \}$$

↑ Not the definition of $\mathcal{C}[\![\cdot]\!]$ but rather a very important theorem.

This is what matters about a command!!

<u>Remark:</u> $c_1 \sim c_2$ iff $\mathcal{C}[\![c_1]\!] = \mathcal{C}[\![c_2]\!]$

other style - tell you at an abstract mathematical level what the meaning of the code is.

<u>Denotational Semantics</u>

Denotation of commands is a binary relation on states
Denotation of Aexp is a relation between states & nums
Denotation of Bexp is a relation between states & Bools.

It will be a theorem that we prove which is that these are actually functional denotational semantics will be defined by structural induction on commands. — which we could not do for $\rightarrow$.

we did not have a structural induction of $\rightarrow_a^*$ to σ'

$\mathcal{A}[\![a]\!] \subseteq \Sigma \times Num$, as a matter of fact, the relation will be a total function, so we also could say:

$\mathcal{A}[\![a]\!] : \Sigma \rightarrow Num$.

$\mathcal{A}[\![\cdot]\!] : Aexp \rightarrow (\Sigma \rightarrow Num)$

Defn By structural induction on Aexp:

$(\mathcal{A}[\![n]\!])(\sigma) = n$.

$\mathcal{A}[\![X]\!]\sigma = \sigma(X)$

$\mathcal{A}[\![a_0 + a_1]\!]\sigma = \mathcal{A}[\![a_0]\!]\sigma + \mathcal{A}[\![a_1]\!]\sigma$

($+$ on the left: symbol character in aexps; $+$ on the right: english name for sum function.)

$\mathcal{A}[\![n]\!] = \lambda\sigma.\, n$.

$\mathcal{A}[\![X]\!] = \lambda\sigma.\, \sigma(X)$

$\mathcal{A}[\![a_0 + a_1]\!] = \lambda\sigma.\, \mathcal{A}[\![a_0]\!]\sigma + \mathcal{A}[\![a_1]\!]\sigma$.

$= \mathcal{A}[\![a_0]\!] \oplus \mathcal{A}[\![a_1]\!]$

technically speaking this is a different addition

6.044 L12
10/9/91
p2

~~Skip~~ Bexps same thing
~~except at Prec Cost a_0~~

$\mathbb{T} = \{\text{true}, \text{false}\}$

$$\mathcal{B}[\![a_0 = a_1]\!]\sigma = \begin{cases} \text{true} & \text{if } \mathcal{A}[\![a_0]\!]\sigma = \mathcal{A}[\![a_1]\!]\sigma \\ \text{false} & \text{ow} \end{cases}$$

$$\mathcal{B}[\![\cdot]\!] : Bexp \to (\Sigma \to \mathbb{T})$$

Remark: we will learn that likewise.

$$\mathcal{C}[\![\cdot]\!] : Com \to [\Sigma \to_{partial} \Sigma]$$

From the form of the defintion it will not be obvious that it is a function.

Actually will define by struct. ind.

$$\mathcal{C}[\![\cdot]\!] : Com \to Pow(\Sigma \times \Sigma).$$

$$\mathcal{C}[\![skip]\!] ::= \{(\sigma, \sigma) \mid \sigma \in \Sigma\}.$$

$$\mathcal{C}[\![X := a]\!] ::= \{(\sigma, \sigma[n/X]) \mid n = \mathcal{A}[\![a]\!]\sigma\}, \sigma \in \Sigma\}$$

$$= \{(\sigma, \sigma[\mathcal{A}[\![a]\!]\sigma / X]) \mid \sigma \in \Sigma\}$$

$$\mathcal{C}[\![c_0 ; c_1]\!] ::= \{(\sigma, \sigma') \mid \exists \sigma''. \ (\sigma, \sigma'') \in \mathcal{C}[\![c_0]\!] \text{ and } (\sigma'', \sigma') \in \mathcal{C}[\![c_1]\!]\}$$

Clearly closely related to "evals to" rule but different.

$$= \mathcal{C}[\![c_1]\!] \circ \mathcal{C}[\![c_0]\!]$$

↖ relational composition, Ch 1.

(notice the.

6.044L/13 Call Fix it
253 - 49 48

p/ 10/11

2 blown. room middle row bulbs on rightmost & 2nd from rightmost sets of bulbs.

$\mathcal{A}[\![\cdot]\!] : Aexp \to (\Sigma \to Num)$

$\mathcal{B}[\![\cdot]\!] : Bexp \to (\Sigma \to T)$

$\mathcal{C}[\![\cdot]\!] : Com \to (\Sigma \rightharpoonup \Sigma)$ ← Partial function

~~Aming for Termi~~

Def by structural induction on syntax of Aexp, Bexp, Com.

Aiming For

Theorem ("Equivalence" of operational (evaluation) semantics and denotational sem.

$\mathcal{A}[\![a]\!]\sigma = n$ iff $\langle a, \sigma\rangle \to n$

$\mathcal{B}[\![b]\!]\sigma = t$ iff $\langle b, \sigma\rangle \to t$

$\mathcal{C}[\![c]\!]\sigma = \sigma'$ iff $\langle c, \sigma\rangle \to \sigma'$.

10 people
good long motivational chat.

Fact:

$$\mathcal{C}[\![\text{while } b \text{ do } c]\!]\sigma \stackrel{?}{=} \begin{cases} \sigma & \text{if } \mathcal{B}[\![b]\!]\sigma = \text{false}, \\ \mathcal{C}[\![c; \text{while } b \text{ do } c]\!]\sigma & \mathcal{B}[\![b]\!]\sigma = \text{true} \end{cases}$$

$$= \mathcal{C}[\![\text{if } b \text{ then } (c; \text{while } b \text{ do } c) \text{ else skip}]\!]\sigma$$

$$= \begin{cases} \sigma & \text{if } \mathcal{B}[\![b]\!]\sigma = \text{false}, \\ (\mathcal{C}[\![\text{while } b \text{ do } c]\!] \circ \mathcal{C}[\![c]\!])\sigma & \text{otherwise}. \end{cases}$$

This cannot serve as defn of while
why not? it is not a structural definition.

This cannot be a Definition of While, it is merely a constraint!

let φ be $\mathcal{C}[\![\text{while } b \text{ do } c]\!]$ φ is a partial function

last eqn can be understood as saying

$\varphi(\sigma) = \ldots \varphi \ldots \sigma \ldots$
x x

you can understand this as a constraint or equation

is there a φ at all which satisfies eqn
are there φ more than 1
if so how to pick 1.

Say $f : \mathbb{N} \to \mathbb{N}$

(1) $f(x) = 3 \times f(x) + 1$
No such f!

(2) $f(x) = 3 \times f(x)$
$f(x) = 0$ satisfies it and only that f.
- there is a unique f.

(3) $f(x) = (f(x))^2$
f could be any 0-1 valued function!
there is a unique smallest one in the usual order $0 < 1$;
$f(x) \equiv 0$

These will all be true facts but none can serve as defn.

6.044 L13
p 2 10/11.

Define ~~$\Gamma(\cdot)(\Sigma \to \Sigma) \to$~~

suppose we had some particular while b do c.

$\Gamma_{b,c} : (\Sigma \to \Sigma) \to (\Sigma \to \Sigma)$

by the rule

$$\Gamma(\varphi)\,\sigma =_{df} \begin{cases} \sigma & \text{if } \mathcal{B}[\![b]\!] = \text{false} \\ \varphi\,(\mathcal{C}[\![c]\!]\,\sigma) & \text{o.w.} \end{cases}$$

$(\varphi \circ \mathcal{C}[\![c]\!])\,\sigma$

__Motivation:__

$\Gamma(\varphi)\sigma$ ≅ the text of $\mathcal{C}[\![\text{while } b \text{ do } c]\!]\,\sigma$ with φ plugged in for $\mathcal{C}[\![\text{while } b \text{ do } c]\!]$

$$\Gamma(\mathcal{C}[\![\text{while } b \text{ do } c]\!]) = \begin{cases} b & \text{if } \mathcal{B}[\![b]\!]\sigma = \text{false}, \\ (\mathcal{C}[\![\text{while } b \text{ do } c]\!] \circ \mathcal{C}[\![c]\!])\,\sigma & \text{ow} \end{cases}$$

So $\mathcal{C}[\![\text{while } b \text{ do } c]\!]$ will be a fixed point of $\Gamma_{b,c}$ by def of $\Gamma_{b,c}$.

in general things like gamma can have many fixed points!!

Γ is a total function on partial functions.

Now Define $\mathcal{C}[\![\text{while } b \text{ do } c]\!] = \varphi$ by the condition that φ is the __least__ solution to the equation

$$\varphi = \Gamma_{b,c}(\varphi).$$

Problems (1) is there a φ satisfying the equation?
(2) least in what sense?
(3) is there a unique least one?

$C[\![\text{while } b \text{ do } c]\!] =_{df} \text{fix}(\Gamma_{b,c})$.

where $\text{fix}(\Gamma_{b,c})$ says pick the least fixed point of $\Gamma_{b,c}$.

we must answer all of the problem. ← pulled out of a hat.

observe that $\Gamma_{b,c}$ is equal to $\hat{R}$ for a set of rules R (read about & we have assigned).

$\therefore$ $\hat{R}$ has a least fixed point under set inclusion, namely

$$\bigcup_{n \geq 1} \hat{R}^{(n)}(\emptyset)$$

verbal argument arg:

$$\hat{R}\left(\bigcup_{n \geq 1} \hat{R}^{(n)}(\emptyset)\right) = \bigcup_{n \geq 1} \hat{R}^{(n)}(\emptyset)$$

The proofs are not very hard.
the hard part is knowing what to prove.

10/16/91
p1
6.044L14

$\varphi : \Sigma \to \Sigma$

$$\Gamma_{B,c_0}(\varphi)(\sigma) = \begin{cases} \sigma & \text{if } \mathcal{B}[\![b]\!]\sigma = \text{false}, \\ \varphi(\mathcal{C}[\![c_0]\!]\sigma) & \text{o.w.} \end{cases}$$

our definition →

$$\mathcal{C}[\![\text{while } b \text{ do } c_0]\!] = \text{fix}\ \Gamma$$

commentary & motivation →

$$\left(= \mathcal{C}[\![\text{if } b \text{ then } c_0; \text{while } b \text{ do } c_0 \text{ else skip}]\!] \right)$$

informal – mixture of syntax ~~and~~ semantics.

$$\Gamma(\varphi) = \mathcal{C}[\![\text{if } b \text{ then } c_0; \varphi \text{ else skip}]\!]$$

Γ can be understood as a set theoretic mapping!

$\Gamma_{b,c} = \hat{R}$ looking $\hat{R}\left(\bigcup_{n>0} \hat{R}^{(n)}(\emptyset)\right) = \bigcup_{n>0} \hat{R}^{(n)}(\emptyset)$

Aside: $f : A \to A$

~~$f^{(0)}(x) =_{df} x$~~

$f^{0} =_{df} id_A$

$f^{(n+1)} = f \circ f^{(n)}$

$$\Gamma^{(0)}(\emptyset) = \emptyset = \mathcal{C}[\![\text{"diverge"}]\!]$$

$\Gamma^{(1)}(\emptyset) =$ ~~$\lambda\sigma.\ \sigma$ if $\mathcal{B}[\![b]\!]\sigma = \text{false}$~~

$$\{(\sigma,\sigma) \mid \mathcal{B}[\![b]\!]\sigma = \text{false}\} = \mathcal{C}[\![\text{if } b \text{ then } \text{diverge} \text{ else skip}]\!]$$

$$\Gamma^{(2)}(\emptyset) = \Gamma(\ \ ''\ \ ) = \mathcal{C}[\![\text{if } b \text{ then } c_0; (\text{if } b \text{ then diverge else skip}) \text{ else skip}]\!]$$

$\Gamma^{(n+1)} = \mathcal{C}[\![$ if b then c_0 ; (if b then c_0 ; (...) else skip) else skip $]\!]$

$fix(\Gamma) = \bigcup_{n \geq 0} \Gamma^{(n)}$

$\Gamma^{(n+1)}(\emptyset)\sigma = \Gamma^{(n)}(\emptyset)\ \sigma$ if $\sigma \in dom(\Gamma^{(n)}(\emptyset))$

"In the limit we have explained the meaning of while in terms of conditionals"

While is an abbreviation for an infinite pgm.

Thm: (Equivalence of operational & denotational semantics)

$\mathcal{C}[\![c]\!]\sigma = \sigma'$ iff $\langle c, \sigma \rangle \to \sigma'$

$\mathcal{A}[\![a]\!]\sigma = n$ iff $\langle a, \sigma \rangle \to n$

$\mathcal{B}[\![b]\!]\sigma = t$ iff $\langle b, \sigma \rangle \to t$

give hint for hw need to prove all 3 simultaneously.

V. imp. Pv thm for Aexps
Pv thm for Bexps
Pv thm for Com.

in general must take conj of 3 thms as ind hypothesis

pf $\mathcal{A}$ & $\mathcal{B}$ by structural induction.

10/16/01
p2
6.044

For $\mathcal{C}$: (i) $\langle c,\sigma\rangle \to \sigma' \Rightarrow \mathcal{C}[\![c]\!]\sigma = \sigma'$

by <u>rule induction.</u>

Case (1. while true) so $c \equiv$ while b do c_0

and

$$\frac{\langle b,\sigma\rangle \to \text{true},\ \langle c_0,\sigma\rangle \to \sigma'',\ \langle c,\sigma''\rangle \to \sigma'}{\langle c,\sigma\rangle \to \sigma'}$$

By induction $\mathcal{C}[\![c_0]\!]\sigma = \sigma''$, $\mathcal{C}[\![c]\!]\sigma'' = \sigma'$
$\mathcal{B}[\![b]\!]\sigma = \text{true}$,

so $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}[\![\text{if } b \text{ then } c_0; c \text{ else skip}]\!]\sigma$

proved already follows because $\Gamma(\mathcal{C}[\![\text{while}]\!]) = \mathcal{C}[\![\text{while}]\!]$.

$$=_{\text{df}_{\mathcal{C}}} \begin{cases} \mathcal{C}[\![c_0;c]\!]\sigma & \mathcal{B}[\![b]\!] = \text{true} \\ \sigma & \text{o.w.} \end{cases}$$

$= \mathcal{C}[\![c_0;c]\!]\sigma$

$=_{\text{df}_{\mathcal{C}}} \mathcal{C}[\![c]\!](\mathcal{C}[\![c_0]\!]\sigma)$

$= \mathcal{C}[\![c]\!](\sigma'')$

$= \sigma'$

(d) $\mathcal{C}[\![c]\!]\sigma = \sigma' \Rightarrow \langle c, \sigma\rangle \to \sigma'$

pf by structural induction.

2. ($c \equiv$ while b do c_0)

For this subcase have $\mathcal{C}[\![c]\!] = \bigcup_{n \geq 0} \Gamma_{b,c}^{(n)}(\phi)$

prove ~~that~~ by induction on n, that

$$\Gamma^{(n)}(\phi)\sigma = \sigma' \Rightarrow \langle c, \sigma\rangle \to \sigma' \quad (*)$$

Then

$\mathcal{C}[\![c]\!]\sigma = \sigma' \Rightarrow \Gamma^{(n)}(\phi)\sigma = \sigma'$ for some $n \geq 0$.

$= \left(\bigcup_{n \geq 0} \Gamma^{(n)}(\phi)\right)\sigma = \sigma'$ $\Downarrow$ by $(*)$

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

G1044 Send Forum msg Skip 5.5
p1
10/18 CPO's
Hoare Logic.

least fixed points:

$\Gamma_{\text{while } b \text{ do } c} = \hat{R}$

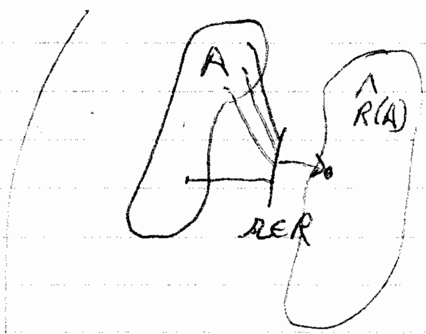
$\bigcup_{n\geq 0}\left(\hat{R}^{(n)}(\emptyset)\right)$ is a fixed point, in fact a least fixed pt. of $\hat{R}$.

monotone Lemma: $A \subseteq B \Rightarrow \hat{R}(A) \subseteq \hat{R}(B)$

Continuity Lemma: If $B_1 \subseteq B_3 \subseteq B_2 \subseteq \ldots$ Let $B = \bigcup_{n\geq 0} B_n$

Then also $\hat{R}(B_0) \subseteq \hat{R}(B_2) \subseteq \ldots$

~~and~~ ~~$\hat{R}$~~ $\hat{R}\left(\bigcup_{n\geq 0} B_n\right) = \bigcup_{n\geq 0} \hat{R}(B_n)$



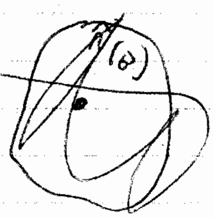
rigorous — formal
↑ for machine

[⊇]

pf: $B_n \subset B$ so $\hat{R}(B_n) \subset \hat{R}(B)$ by monotone

∴ $\bigcup_{n\geq 0} \hat{R}(B_n) \subset \hat{R}(B)$

[⊆]. ~~suppose~~



Suppose $p \in \hat{R}(B)$. we want to show $p \in \hat{R}(B_n)$ for some $n \geq 0$.

Say $p \in \hat{R}(B)$ because of rule $r = \{p_1, p_2, p_3 / p\}$

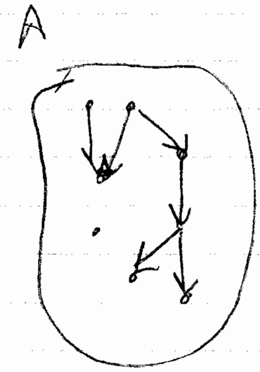
Where $p_i \in B$ thus $p_i \in B_{n_i}$ for $i = 1, 2, 3$ by def. of B.

$\therefore p_1, p_2, p_3 \in B_{\max\{n_1, n_2, n_3\}}$

$\therefore p \in \hat{R}(B_{\max\{n_1, n_2, n_3\}})$ — $\max\{n_1, n_2, n_3\} = n$

Dana Scott:

Def: Partial Order: $A, \sqsubseteq$



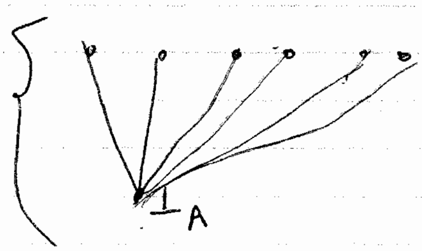
(0) ~~always have~~

(1) No loops of positive length!

DAG. Directed acyclic graph.

$p \sqsubseteq q$ iff there is a path from p to q.
iff $p \to^* q$.

$A_\perp$



partial order on A: discrete partial
$p \sqsubseteq_{dis} q$ iff $p = q$.

$A_\perp =_{df} A \cup \{\perp_A\}$ where $\perp_A \notin A$
$p \leq_{A_\perp} q$ iff $p = q$ or $p = \perp_A$

6.044L

pd 10/18

<u>Complete partial orders</u>

if $b_0 \sqsubseteq b_1 \sqsubseteq b_2 \sqsubseteq \cdots$

the ~~lub {b_i} exists.~~ lub $\{b_i \mid i \geq 0\}$ exists

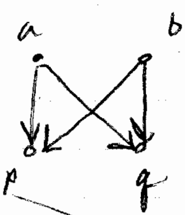
$$\bigsqcup_{n \geq 0} b_i$$

$p \subset A$ an upper bound on p is an elt of A bigger than every elt of p.

every set of Real numbers that has a bound at all has a lub.

every subset of $(1, 2)$ has an lub.

think about Q !!



p & q don't have a lub

o k as a cpo

is not a cpo.

$f: A \to B$

monotone $a_1 \sqsubseteq_A a_2 \Rightarrow f(a_1) \sqsubseteq_B f(a_2)$

continuity monotone and:

if $b_0 \sqsubseteq b_1 \sqsubseteq b_2 \sqsubseteq \cdots$

$$f\left(\bigsqcup_{i \geq 0} b_i\right) = \bigsqcup_{i \geq 0} f(b_i)$$

("f preserves lubs")

~~if ⊥ les' $\bigsqcup_{n \geq 0} f^{(n)}(\bot)$~~

if $\bot$ is least element of A $f : A \to A$ is continuous

then $fix\ f = \bigsqcup_{n \geq 0} f^{(n)}(\bot)$

send Typo Mail

send Ken Ruda Mail

6.044L (Larry is Wimmy's mentor)
p1
10/30/91

Assn:

$S ::= a_1 = a_2 \mid a_1 \leq a_2 \mid S_1 \wedge S_2 \mid S_1 \vee S_2 \mid \neg S \mid \forall i. S \mid \exists i. S$

$a ::= X \mid n \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \mid i$

↖ integer variables

$gcd(m, N) = gcd(m', N')$

Talk about expressive power of Assn, how to say
$gcd(m, N) = gcd(m', N')$

$\exists i.\ i = gcd(m, N) \wedge i = gcd(m', N')$

Not an Aexp.

$i \mid m \wedge i \mid N \wedge \forall k.\ k \mid m \wedge k \mid N$ ~~implies~~ $\Rightarrow k \leq i$

$\exists j, i \times j = m$

$\in$ Assn

$\neg(k \mid m \wedge k \mid N) \vee k \leq i$

$\{A\}\ c\ \{B\}$ for all states σ satisfying A if you do c you end up in a state satisfying B.

$I \models i = j$
$\sigma \models X < Y$

$I:$ Intvar $\rightarrow$ Num

if we write ~~$\forall X.$~~ it is dangerous?! are quantifying.

$\sigma, I \models X = i$ Winskel uses $\sigma \models^I S$ Ternary relation

$\models$ is a relation between a state Interpretation, & assertion.

Def: $Av[\![\cdot]\!] : Aexp \to (\text{interpretation} \to (\Sigma \to Num))$

$(Av[\![a]\!](I))(\sigma) = n$

$Av[\![X]\!] I \sigma = \sigma(X)$
$Av[\![i]\!] I \sigma = I(i)$

$Av[\![a_1 + a_2]\!] I \sigma = Av[\![a_1]\!] I \sigma + Av[\![a_2]\!] I \sigma.$

<u>Lemma:</u> $Av[\![a]\!] I \sigma = A[\![a]\!] \sigma$
if $a \in Aexp$ w/o integer variables.

$\sigma \models^I a_1 = a_2$ iff $Av[\![a_1]\!] I \sigma = Av[\![a_2]\!] I \sigma.$

$\sigma \models^I \forall i. S$ iff $\sigma \models^{I'} S$ for <u>all</u> I' which differ from I only at i
That is, $I'(j) = I(j)$ for $j \neq i$
don't care $I'(i)$

notation $\models^I S$
notation for "$\sigma \models^I S$ for all σ"

likewise $\models S$ notation for "$\models^I S$ for all I"

$\models ((A \vee B) \Rightarrow \neg(\neg A \wedge \neg B)) \wedge (A \vee B \Leftarrow \neg(\neg A \wedge \neg B))$

First-order arithmetic

Remark: (Lemma)
$\sigma \models^I \forall i. S$ iff $\sigma \models^I S[n/i]$ for all $n \in Num$

$S[n/i]$ is not a subexpression not a structural Induction definition.

6.044
10/21/91
p1.

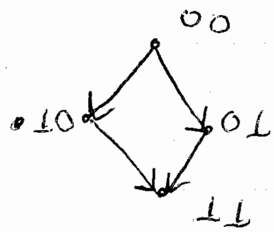
more on cpo's.

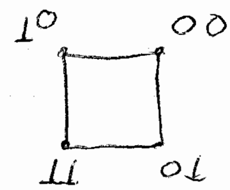
<u>Lemma</u> If A, B are cpo's, so is $(A\times B, \sqsubseteq_{A\times B})$ where $\langle a_1, b_1\rangle \sqsubseteq_{A\times B} \langle a_2, b_2\rangle$

iff $a_1 \sqsubseteq_A a_2$ and $b_1 \sqsubseteq_B b_2$

Say $A = \{0\}_\perp$

$A\times A =$





and so is $A\to B =_{df}$ the continuous functions from A to B.

where $f \sqsubseteq_{A\to B} g$ iff $f(a) \sqsubseteq_B g(a)$ for all $a\in A$.

must verify $\sqsubseteq_{A\to B}$ is a po & it is complete.

pf(1) $\sqsubseteq_{A\times B}$ is a p.o.

e.g. if $\langle a_1, b_1\rangle \sqsubseteq_{A\times B} \langle a_2, b_2\rangle$

and $\langle a_2, b_2\rangle \sqsubseteq \langle a_3, b_3\rangle$,

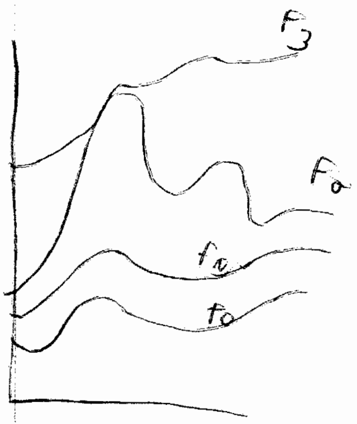
is $\langle a_1, b_1\rangle \sqsubseteq \langle a_3, b_3\rangle$?

(2) lub's exist $\langle a_i, b_i\rangle \sqsubseteq \langle a_{i+1}, b_{i+1}\rangle$ for $i \geq 0$,

then $\bigsqcup_i \langle a_i, b_i\rangle = \langle \bigsqcup_i a_i, \bigsqcup_j b_j\rangle$

pf show $\sqsupseteq$ & $\sqsubseteq$, so = follows by antisymmetry.

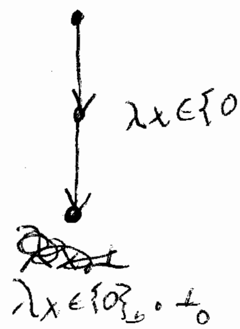
note that $\bigsqcup_{i \geq 0} f_i$ where $f_0 \sqsubseteq f_1 \sqsubseteq f_2$.



so $\left(\bigsqcup_{i \geq 0}{}_{A \to B} f_i\right)(a)$ where $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \cdots$

$= \bigsqcup_{i \geq 0}{}_B \left(f_i(a)\right)$

$\{0\}_\perp \to \{0\}_\perp$



<u>Some facts:</u> (1) Composition preserves continuity:

if $f \in A \to B$ and $g \in B \to C$

then so is $(g \circ f): A \to C$

<u>First proof:</u> $(g \circ f)\left(\bigsqcup a_i\right) \stackrel{?}{=} \bigsqcup_i (g \circ f)(a_i)$

$\parallel$

$g\left(f\left(\bigsqcup_i a_i\right)\right)$

$\parallel$

$g\left(\bigsqcup_i f\left(\bigsqcup_i a_i\right)\right)$

$\parallel$

$\bigsqcup_i g\left(f(a_i)\right)$.

(2) $f: A \times B \to C$ is continuous iff $f_a : B \to C$ is continuous for all $a \in A$
$f^b : A \to C$ is continuous for all $b \in B$

$f_a : \lambda x \in B.\ f(a, x)$ $\quad f^b = \lambda y \in A.\ f(y, b)$.

6.044
10/1/91
p2

$A ::= a_1 = a_2 \mid a_1 \le a_2 \mid \underline{true} \mid \underline{false} \mid A_1 \wedge A_2 \ (\vee, \Rightarrow) \mid \neg A \mid \exists i.\, A \mid \forall i.\, A$

$\sigma \models^I A$

logic: "$A \equiv B$" abbrev "$(A \to B) \wedge (B \Rightarrow A)$"

$\models \neg(\forall i.\, A) = \exists i.\, \neg A$

partial correctness assertion $s \notin Assn$

Dyn Assn:

$D ::= \ldots \mid D_1 \wedge D_2 \mid \ldots \mid \{D_1\}\, c\, \{D_2\}$

basic idea:

$\{true\}\, c\, \{X=1\}$

$\{\models \{true\}\, c\, \{X=1\}\}\ c_0; c\ \{X=1\}$

$\sigma \models^I D$ defined as $\sigma \models^I A$, except for new case

$\sigma \models^I \{D_1\}\, c\, \{D_2\}$ iff [if $\sigma \models^I D_1$ + if $\mathcal{C}[\![c]\!]\sigma$ is defined, then $\mathcal{C}[\![c]\!]\sigma \models^I D_2$]

$\sigma \models^I D_1$ implies $\mathcal{C}[\![c]\!]\sigma \models D_2$

$\mathcal{C}[\![c]\!]: \Sigma \to \Sigma_\perp$

$\perp \models^I A \qquad \perp \models D$

$\models \{D_1\}\, c\, \{D_2\} \equiv (D_1 \Rightarrow \{true\}\, c\, \{D_2\})$

If D_1 holds then after doing c D_2 holds

weakest precondition of D_2 under c.

If D_1 holds then no matter what, after done c D_2 holds.

Weakest Precondition Spec.

~~$\sigma \models^I wp(c, D)$~~

$\sigma \models^S w(c, D_\lambda)$ iff $\mathcal{C}[\![c]\!]\sigma \models^I D_\lambda$

Assignment axiom:

$\{A[a'/x]\}\ X := a'\ \{A\}$

Lemma: $\models A[a'/x] \equiv w(X := a', A)$

this only works because we're dealing w/ A, bad things would happen w/ D.

$((\{\{X = Y\}\ X := X+2\ \{\text{even}(X)\}\}\ \text{if } \text{prime}(x) \text{ then } x := 2 \text{ else } x := 2\ \{X = Y\})[3/X]$

Def. "A_n" assertion language is <u>expressive</u> iff for every assertion A in the language, and every IMP command c, there is an assertion $W_{A,c}$

such that $\models W_{A,c} \equiv W(c, A)$

Trivially, Dyn Assn are expressive
$w(c, A) = \{\text{true}\}\ c\ \{A\}$

<u>Deep Fact:</u> Assn <u>is expressive</u> (requires some very tricky proof).

6.044
p1
11/4/91

~~Notation: "$\mathcal{E}_c$"~~

Lemma: Assn is expressive for IMP.

That is, for all $c \in$ IMP, $B \in$ Assn (not Dyn Assn),
there is an assertion $W(c,B) \in$ Assn
such that $W(c,B)$
expresses the weakest precondition of B after c.

$\sigma \models^I W(c,B)$ iff $\mathcal{C}[\![c]\!]\sigma \models^I B$.

$\models W(c,B) \equiv \{true\}\, c\, \{B\}$

In fact, we will show a procedure to translate $\langle c, B \rangle$ into an assertion $W(c,B)$

Cor: For every $D \in$ Dyn Assn, there is an $A \in$ Assn such that $\models A \equiv D$

pf: replace innermost D, $\{A_1\}\, c\, \{A_2\}$ where A_1, A_2 ordinary assertions.

$$A_1 \Rightarrow W(c, A_2)$$

Spiel on completeness & soundness.

pf: By structural induction on c:

$W(\text{skip}, B) ::= B$

$W(X:=a, B) ::= B[a/X]$

$W(\text{if } b \text{ then } c_0 \text{ else } c_1, B) ::= (b \Rightarrow W(c_0, B)) \wedge (\neg b \rightarrow W(c_1, B))$

$W((c_0 ; c_1), B) ::= W(c_0, W(c_1, B))$

Let $c \equiv \text{while } b \text{ do } c_0$

$W(c, B)$?

Remember: $\sigma_0 \models^I W(c, B)$ should hold
iff $\mathcal{C}[\![c]\!]\sigma_0 \models^I B$.

(stop writing I, it's fixed, should be there)

iff there is a $k \geq 0$ and $\sigma_0, \sigma_1, \ldots, \sigma_k$ s.t.

(1) $\sigma_k \models \neg b$

(2) $\mathcal{C}[\![c_0]\!]\sigma_i = \sigma_{i+1}$ for $0 \leq i < k$ (vacuous when $k=0$)

(3) $\sigma_i \models b$

idea, how to express this as an assertion.

digression

Gödel – numbering,

strategy think of a state as a number
+ think of a sequence of states as a number.

Suppose that $loc(c) = \{X_1, X_2\}$

Cantor – infinities come in different sizes.

represent a state σ for c's purposes as pair $\langle \sigma(X_1), \sigma(X_2) \rangle$.

6.044J
10/21/91
p1

$f: A \times B \to C$ is continuous if it is continuous in each argument. for cpo's.

$\Sigma \to \Sigma$ is a cpo under $\subseteq$

New Topic:

(Floyd –) Hoare Logic

We want methods to prove things about programs.

$\{\quad\}$ Euclid $\{\quad\}$

precondition   postcondition.

A.

$\{M, N > 0$ and $X = gcd(M,N)\}$ Euclid $\{M = X\}$

$X \neq M$ or N

while $M \neq N$ do

$($if $M \leq N$ then $N := N - M$ else $M := M - N)$

$$\frac{\{A \wedge M \neq N\} \text{ if } M \leq N \text{ then } N := N-M \text{ else } M := M-N \{A\}}{\{A\} \text{ Euclid } \{A \wedge M = N\}}$$

Find an A st
$M, N > 0$ and $X = gcd(M,N) \Rightarrow A$
$+ A \wedge M=N \Rightarrow M = X$.

~~$A = M, N > 0$~~

6.044J
10/23
p1

Hoare Logic: Assertions: A, B, C, A', B'

(1) $\{A\}\ \text{skip}\ \{A\}$

(2) $\{A[a/x]\}\ x := a\ \{A\}$

(3) $$\frac{\{A\}\ c_1\ \{B\},\quad \{B\}\ c_2\ \{C\}}{\{A\}\ c_1 ; c_2\ \{C\}}$$

(4) $$\frac{\{A \text{ and } b\}\ c_1\ \{B\} \qquad \{A \wedge \neg b\}\ c_2\ \{B\}}{\{A\}\ \text{if } b \text{ then } c_1 \text{ else } c_2\ \{B\}}$$

(5) $$\frac{\{A \wedge b\}\ c\ \{A\}}{\{A\}\ \text{while } b \text{ do } c\ \{A \wedge \neg b\}}$$

(6) $$\frac{A \text{ implies } A',\quad B' \text{ implies } B,\quad \{A'\}\ c\ \{B'\}}{\{A\}\ c\ \{B\}}$$

Euclid ::= while $m \neq N$ do if $m \leq N$ then $N := N - m$
else $m := m - N$

$\{m' = m,\ N' = N \text{ and } m, N > 0\}$ Euclid $\{\gcd(m', N') = m\}$
where $m', N' \in \text{loc}(\text{Euclid})$

A is true or false in σ.

$\sigma \models A$

$\sigma \models X = X$ for all σ

$\models X = X$ is valid

$\sigma[3/X][2/Y] \models X = Y + 1$

$\sigma[3/X][3/Y] \not\models X = Y + 1$

$\models \neg(X = Y + 1)$

$\{A\}\ c\ \{B\}$ holds (is true)

iff$_{def}$ for all σ:
$\sigma \models A$ implies $\mathcal{C}[\![c]\!]\sigma \models B$
(if this exists)

Thus $\{true\}$ while true do c $\{false\}$

$\{true\}\ c\ \{false\}$ iff $c \sim$ while true do skip
Ω

$\sigma \models b$ $::=_{df}$ $\mathcal{B}[\![b]\!]\sigma = true$

(Satisfies)

pf: Soundness of S
because $\mathcal{C}[\![\text{while } b \text{ do } c]\!]\sigma = \sigma'$ iff [($\sigma \models \neg b$ and $\sigma = \sigma'$) or
there are $\sigma = \sigma_0, \sigma_1, \ldots, \sigma_n = \sigma'$ $(n \geq 1)$
and $\sigma_{i+1} = \mathcal{C}[\![c]\!]\sigma_i$
and $\sigma_i \models b$ for $0 \leq i < n$
$\sigma_n \models \neg b$.]

Substitution Lemma:

$\sigma \models A[a/x]$ iff $\sigma[\mathcal{A}[\![a]\!]\sigma / x] \models A$
$\qquad = \mathcal{C}[\![x := a]\!]\sigma$

if $\sigma \models A[a/x]$ then $\mathcal{C}[\![x := a]\!]\sigma \models A$

$\{A[a/x]\}\ x := a\ \{A\}$

6.044
10/23
p2

Pre ... Post

$\{m' = m, N' = N$ and $m, N > 0\}$ Euclid $\{m = gcd(m', N')\}$

Pf: Find an A s.t.

(1) Pre implies A ✓

(2) (A and $m = N$) imply Post ✓

(3) $\{A\}$ Euclid $\{A \wedge m = N\}$

Magic loop-invariant:

$A ::= m, N > 0$ and $gcd(m', N') = gcd(m, N)$

$\{A \wedge m \neq N\}$ if $m \leq N$ then $N ::= N - m$ else $m ::= m - N$ $\{A\}$

let B be $(m \leq N$ and $A[N-m/N]) or (m > N$ and $A[m-N/m])$.

~~and show (4) $\{A \wedge m \neq N\}$~~

(1) $\{A\}\ \mathrm{skip}\ \{A\}$ (skip) { }

(2) $\{A[a/x]\}\ x := a\ \{A\}$ (assign) { }

(3) $$\frac{\{A\}\ c_1\ \{B\},\quad \{B\}\ c_2\ \{C\}}{\{A\}\ c_1; c_2\ \{C\}}$$ (seq) { }

(4) $$\frac{\{B \wedge b\}\ c_1\ \{A\}\quad \{B \wedge \neg b\}\ c_2\ \{A\}}{\{B\}\ \mathrm{if}\ b\ \mathrm{then}\ c_1\ \mathrm{else}\ c_2\ \{A\}}$$ (if) { }

(5) $$\frac{\{A \wedge b\}\ c\ \{A\}}{\{A\}\ \mathrm{while}\ b\ \mathrm{do}\ c\ \{A \wedge \neg b\}}$$ (while) { }

(6) $$\frac{A \text{ implies } A',\quad B' \text{ implies } B\quad \{A'\}\ c\ \{B'\}}{\{A\}\ c\ \{B\}}$$ (weakening)

Recap

Euclid ::= While $M \neq N$ do
if $M \leq N$ then $N := N - M$
else $M := M - N$

Goal $\{M' = M, N' = N \text{ and } M, N > 0\}$ Euclid $\{gcd(M', N') = M\}$ (Pre = $\{M' = M, N' = N \text{ and } M, N > 0\}$; Post = $\{gcd(M', N') = M\}$)
(where $M', N' \notin Loc(Euclid)$).

How do we prove this using Hoare Logic?

Find an A s.t.
(1) Pre implies A
(2) (A and M = N) imply Post } Done wed
(3) $\{A\}$ Euclid $\{A \wedge M = N\}$.

(It should be clear by (weakening) that if I find an A w/ these properties I'm done!)

(Here it is:)

"finding A is magic" $A \equiv M, N > 0$ and $GCD(M', N') = gcd(M, N)$

(Albert argued 1,2 on wed — I won't bother doing again).

To prove (3), find B s.t.

(4) A and $M \neq N$ implies B
~~(5) $\{B$ and $M \leq N\}$ $N := N - M$ $\{A\}$~~
(5) ~~$\{B\}$ if b then M~~
$\{B\}$ if $M \leq N$ then $N := N - M$ else $M := M - N$ $\{A\}$

(it should be clear by (weakening) and (while) that if I prove (4)+(5) then I've proven 3 + so I'm done)

(4) + (5) $\Rightarrow$ (3) by (weakening) + (while)

Finding B is <u>Not</u> Magic!

$B \equiv \; M \le N \;\&\; A[^{N-M}/_N]$
or
$M > N \;\&\; A[^{M-N}/_N]$

pf(5) ~~sub A~~ ~~N-M~~

$\{B \wedge M \le N\}$ implies $A[^{N-M}/_N]$ (obviousley)

$\{A[^{N-M}/_N]\}\; N := N-M \;\{A\}$ (assign)

so $\{B \wedge M \le N\}\; N := N-M \;\{A\}$.

similarly
$\{B \wedge M > N\}\; M := M-N \;\{A\}$.

by (if) $\{B\}$ if $M \le N$ then $N := N-M$ else $M := M-N$ $\{A\}$ ☒

Final step: (uses your knowledge of math)
pf (4). ~~here is where the deep mathematical facts come~~

prove, in any rigorous way, A and $M \ne N$ implies B.

so, suppose $M, N > 0$, $\gcd(M', N') = \gcd(M, N)$ and $M \ne N$.

Case (i) $0 < M \le N$.
well $\gcd(M, N) = \gcd(M, N-M)$.
why? 18.063.

($\gcd(M,N) \equiv$ largest d st $d|M$ + $d|N$.
well $d|M$ eg $d \cdot x = M$.
$d|N$ eg $d \cdot y = N$
note $x < y$.
so $d|N-M$ ($d \cdot (y-x) = N-M$!)

A said $\gcd(M', N') = \gcd(M, N)$
so $\gcd(M', N') = \gcd(M, N-M)$
also $N-M > 0$
so in case (i) $M \le N$ and $A[^{N-M}/_M]$ holds (verbally discuss)

Case (ii) $M > N$ & we have $A[{}^{M-N}/_{N}]$.

pause for Ques.

(verbally argue overall implication as recap)

New Problem

$D \equiv Y:=0;$ (while $X \geq 2$ do $X:=X-2;$ ~~$X:=X-1$~~; $Y:=Y+1$)

what does it do?

(if loads of time argue termination)

$\{X = x' \text{ and } X \geq 0\}\ D\ \{Y = \lfloor x'/2 \rfloor \text{ and } X = rem(x', 2)\}$

(do they know what $\lfloor y \rfloor$ means.
eg $\lfloor 1.1 \rfloor = 1$, $\lfloor 3 \rfloor = 3$ $\lfloor 4.9 \rfloor = 4$.)

Pf:

Find A st

(1) $\{X = x' \text{ and } X \geq 0\}\ Y:=0\ \{A\}$

(2) ~~$\{A\}$ while $X>1$ do $X:=X-1$~~

(2) $\{A \text{ and } X \leq 1\}$ imply $\{Y = \lfloor x'/2 \rfloor \text{ and } X = rem(x', 2)\}$

(3) $\{A\}$ while $X \geq 2$ do $X:=X-2;$ ~~[illegible]~~; $Y:=Y+1$ $\{A \wedge X \leq 1\}$ ~~$\{Y = \lfloor x'/2 \rfloor$ and $X = rem(x', 2)\}$~~

$A \equiv X \geq 0 \wedge x' = 2Y + X$

(1,2) obvious.

(3) show $\{A \wedge X > 1\}\ X:=X-2;\ Y:=Y+1\ \{A\}$

to do: Find (θ) → (arrow to the point between $X:=X-2;$ and $Y:=Y+1$)

here's one $A[{}^{Y+1}/_{Y}]$ ie ~~$Y-1 = \lfloor x'/2 \rfloor$~~ $x' = 2(Y+1) + X$.

Now need A' st $A \wedge X > 1$ implies A'
& $\{A'\}\ X:=X-2\ \{A[{}^{Y+1}/_{Y}]\}$.

good candidate:

$$A' \equiv A[{}^{y+1}/_{y}][{}^{x-2}/_{x}]$$

ie $(x-2) \geq 0$ + $x' = 2(y+1) + (x-2)$

does A imply A'.

sure! as A A' iff $x \geq 2$ + $x' = 2y + 2 + x - 2$
$= 2y + x$

Quiz.

4.4 + $\hat{R}$ stuff

5.1-5.4 Denotational semantics equivalence
+ CPO finger exercises
(+ fixpoint + continuity)

6.044L
p 2
11/4/91

represent a pair $\langle n_1, n_2 \rangle$ as an integer $n = 2^{n_1} \cdot 3^{n_2}$

For ~~any integer n~~

mkpair(n_1, n_2)
left$(\cdot)$
right$(\cdot)$

Represent a sequence of integers (of arbitrary but finite length) as numbers:

Gödel β-function.

<u>Sublemma</u>: There is an assertion $SEQ(i, j, k) \in Assn$

Which "means" j is a sequence whose i^{th} element is k.

For every $n_0, n_1, \ldots, n_k$ $\exists n$ which codes $n_1, \ldots, n_k$

eg $SEQ(7, n, k) = n_7 \ldots$

6.044
p1
11/14/91

$\sigma_0 \models \{true\}$ while b do c_0 $\{B\}$ iff

if $\sigma_0, \sigma_1, \ldots, \sigma_k$ for $k \geq 0$ is st.

(1) $\sigma_k \models \neg b$ and

(2) $\sigma_{i+1} = \mathcal{C}[\![c_0]\!]\sigma_i$ for all i $0 \leq i < k$

(3) $\sigma_i \models b$ for $0 \leq i < k$.

~~2a+b~~
4a+b

Lemma: There is a formula $SEQ \in Assn$, such that for any sequence $n_0, n_1, \ldots, n_k \in \mathbb{Z}$, $k \geq 0$, there is an $n \in \mathbb{Z}$ such that for all $m, m' \in \mathbb{Z}$

$$\models SEQ[n/i][m/j][m'/k] \text{ iff } m' = m_n$$

$(SEQ(i,j,k))$ integer variables $SEQ(z, x, y)$

In other words SEQ means "i is the code of a sequence whose j^{th} element is k"

$\{X_1, X_2\} = loc(\text{while } b \text{ do } c_0)$

$LEFT(i_1, i) = SEQ(i, 0, i_1)$
$RIGHT(i_2, i) = SEQ(i, 1, i_2)$

$SAT_A(l)$ means "l is the code of a state σ s.t. $\sigma \models A$."

A ← Assn

$loc(A) = \{X_1, X_2\}$

$SAT_A(l) ::= \exists i_1, \exists i_2.\ LEFT(i_1, l) \wedge Right(i_2, l) \wedge A[i_1/X_1][i_2/X_2]$

$NEXT_{c_0}(i,j)$ means $\mathcal{C}[\![c_0]\!]$ "state i" = "state j"

$$::= \exists i_1, i_2. \Big(W(c_0, LEFT(x_1, j) \wedge RIGHT(x_2, j)) [i_1/x_1][i_2/x_2] \wedge LEFT(i_1, i) \wedge RIGHT(i_2, i)$$

$$\wedge\ (\neg W(c_0, false))[i_1/x_1][i_2/x_2] \Big)$$

$$\forall k.\ \forall j.\ \forall l.\ \forall l'.$$

$$\Big[k \geq 0 \wedge [SEQ(j,0,l) \wedge LEFT(x_1, l) \wedge RIGHT(x_2, l)] \quad (1)$$

$$\wedge\ [SEQ(j,k,l') \wedge SAT_{\neg b}(l')] \quad (3)$$

$$\wedge \{\forall i.\ (0 \leq i \wedge i < k) \Rightarrow \exists l_1, l_2.\ SEQ(j,i,l_1) \wedge SEQ(j,i+1,l_2) \wedge NEXT_{c_0}(l_1, l_2) \wedge SAT_b(l_1)\} \Big] \quad (2)$$

(need "l_0 is the present state" and l is its code)

$$\Rightarrow SAT_B(l') \quad \text{DONE!!}$$

$|||_{DF}$

$W(c, B)$

6.044 p2
11/6/91 Completeness of Hoare Rules.
p2

Theorem: For $A, B \in Assn$ and $c \in Com$

$$\vdash_{Hoare} \{A\} c \{B\} \text{ iff } \models \{A\} c \{B\}$$

PF: ($\Rightarrow$) Soundness ✓

($\Leftarrow$) By struct. ind. on $c \in Com$

~~$c \equiv skip$~~

$c \equiv X := a$ and $\models \{A\} c \{B\}$.

Then $\models \{A\} c \{B\}$ says $\models A \Rightarrow W(c, B)$

$W(c, B) \in Assn$.

Axiom: $\vdash_{Hoare} \{B[a/X]\} c \{B\}$

also, we know that $B[a/X]$ is an assertion expressing the weakest precondition of B under $X := a$.

$B[a/X] \equiv W(c, B)$ and $\models A \Rightarrow B[a/X]$

By Rule of Consequence.

$\vdash_{Hoare} \{A\} c \{B\}$

6.044J/18.423J
11/8
p1.

$\models \{A\} c \{B\}$ iff $\vdash_{HOARE} \{A\} c \{B\}$

Pf: ($\Rightarrow$) induct on c:

case $c \equiv$ while b do c_0.

Assume $\{A\}\ c\ \{B\}$ (1) is valid. Thus $A \Rightarrow W(c, B)$ is valid
(c Assn)

Sufficient to show that
$\{W(c,B)\} c \{B\}$ (2)
is provable, since $\vdash (2)$ and rule of consequence, yield $\vdash (1)$.

Note that
$\{W(c, B) \wedge b\}\ c_0\ \{W(c, B)\}$ (3)
is valid.

By induction $\vdash_{Hoare}$ (3)

$\therefore$ By while-rule: $\vdash_{Hoare} \{W(c,B)\} c \{W(c,B) \wedge \neg b\}$

Note that $W(c,B) \wedge \neg b \Rightarrow B$ (4) is valid.

$\therefore$ By rule of consequence, $\vdash (2)$ ☒

Hoare Logic Rule of Consequence is problematical.

Individual Hoare rules are checkable by simple ~~pattern~~ matching

$$\frac{A \rightarrow A' \quad , \quad \{A'\}\, c\, \{B\}}{\{A\}\, c\, \{B\}}$$

Want a notion of "proof" for Assn. (Peano System)

$\models \forall i. (\exists i'. 2i' = i) \vee (\exists i. 2i' + 1 = i)$

Axiom system for Assn:

$\forall i, j \quad i+1 = j+1 \rightarrow i = j$

"$\leq$ is a total order"

$\forall i, j, k.\ i \leq j \wedge j \leq k \Rightarrow i \leq k$
$\wedge\ i \leq i$
$\wedge\ i \leq j \wedge j \leq i \Rightarrow i = j$
$\wedge (i \leq j \vee j \leq i)$

$\forall i,\ \neg(i+1 \leq i)$
$\forall i. \exists j.\ j+1 = i$
$\forall i.\ i + 0 = i$
$\forall i, j.\ (i+1) + j = (i+j) + 1$

Rules: Boolean Logic:

$$\frac{A, \quad B}{A \wedge B} \qquad \frac{A}{\neg\neg A}$$

Induction rule:

$$\frac{A[0/i], \quad \forall i.\ A \Rightarrow A[i+1/i]}{\forall i.\ i \geq 0 \Rightarrow A}$$

Thm (Soundness) $\vdash_{Peano} A$ implies $\models A$
Converse $\Leftarrow$ is false

6.044
11/13 p 1

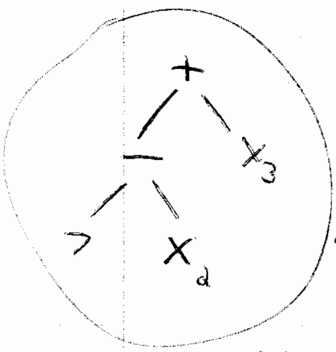
Universal Simulator:

Stored programs

An Imp program which given any IMP program $c \in Com$ and State $\sigma \in \Sigma$, will simulate c on σ.

idea: write one program which understands all programs

Gödel number Com. Namely, list all possible elements of Com in order: Com com_0, com_1, ...



Aexp

$mkpair: \mathbb{N} \times \mathbb{N} \xrightarrow{1-1} \mathbb{N}$

$mkpair(x,y) = 2^x 3^y$

$left: \mathbb{N} \to \mathbb{N}$

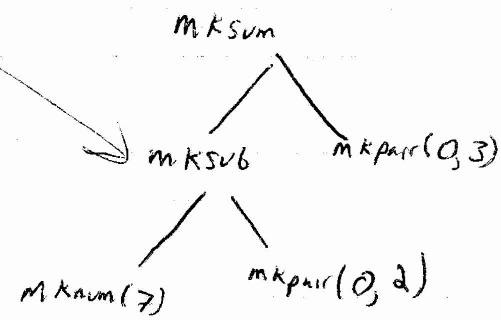
$right: \mathbb{N} \to \mathbb{N}$

$left(mkpair(x,y)) = x$

$right(\quad\quad) = y$

pic $Loc = \{x_1, x_2, x_3, \ldots\}$

0 ⟷ loc
1 ⟷ Num
2 ⟷ +
3 ⟷ −
4 ⟷ X



$mkloc(i) =$ code for x_i
$=_{df} mkpair(0, i)$

$mknum(i) =_{df} mkpair(1, i)$

$mksum(i, j) =$ the code of the Aexp which is the sum of the Aexps coded by i and j
$=_{df} mkpair(2, mkpair(i, j))$

Not all numbers will turn out to code Aexp.

Let any num we don't get that way, by default we'll let it rep 0. Every num can now be regarded as code for an Aexp.

$Sim \in \underset{\sim}{Com}$

ρ be the constant zero state.

$\rho(X_i) = 0$ all $i \geq 0$

$s(n) =$ state w/ $\sigma(X_1) = n$ everywhere else is 0. $n \in Num$. $\rho[n/X_1]$

Want

~~$\mathcal{C}[\![Sim]\!]$~~

$$\mathcal{C}[\![Sim]\!]\big(s(mkpair(n_1, n_2))\big) = s\Big(\big(\mathcal{C}[\![com_{n_1}]\!](s(n_2))\big)\ X_1\Big)$$

Thm Universal Machine Thm:

Sim exists!

Pf: By programming.

$$\frac{\langle b, \sigma\rangle \to true \qquad \langle c_0, \sigma\rangle \to \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \to \sigma'}$$

6.044 p1
11/15/91 Send Email we'll be using IMP no.
p 1 Appendix grammar + evaluation rules.

Spec!!
$$\to \mathcal{C}[\![Sim]\!] s(n) = s\left(\left[\mathcal{C}[\![com_{left(n)}]\!] s(right(n)) \right] (X_1) \right)$$

~~C~~ $com_{left(n)}$ might not halt on $s(right(n))$, in that case Sim won't halt on $s(n)$.

Goal: Prove Gödel's incompleteness thm:

There is not any reasonable way to get an axiomatization of assertions.

<u>Thm</u> There is a Self ∈ Com s.t.

$$\left[\mathcal{C}[\![Self]\!] s(n)\right](X_1) = \left[\mathcal{C}[\![com_n]\!] s(n)\right](X_1)$$

Self: $X_1 := mkpair(X_1, X_1)$
SIM

~~D~~ A ~~program~~ command Not~~Self~~halt ∈ Com is said to be a "non-halting checker" iff

$\mathcal{C}[\![Not\text{~~Self~~}Halt]\!] s(n)$ is defined iff $\mathcal{C}[\![com_n]\!] s(n)$ is undefined.

<u>Thm:</u> There ain't any Nothalt command!

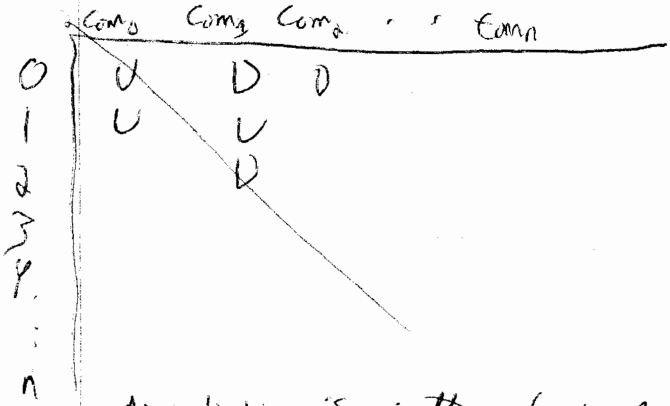
pf: By contradiction suppose there were such a NotHalt command. Now NotHalt $= com_{17}$ say

Thus $\mathcal{C}[\![com_{17}]\!] s(n)$ is defined iff $\mathcal{C}[\![com_n]\!] s(n)$ is undefined, for all n

let $n = 17$, contradiction!

Diagonal argument:

Diagonal Argument



depending on whether Com i on input j is undefined

Not Halt is gotten by looking down diagonal of chart by differing.

Thm: But, there is an assertion Not Halt $A_0 \in$ Assn s.t.

NotHalt A means "$\mathcal{C}[\![com_A]\!] s(\dot{A})$ is undefined".

pf: $W(self, false)\,[^{A}/_{X_1}]\,[^{0}/_{x_j}]$ $j > 1$, $x_j \in loc(self)$

Not Halt A :=

5

6.044L 11/18 p.1

Lemma: "com_i doesn't halt on state $s(i)$" is the meaning of an Assn:

Meaning has nothing to do with "present state" does not have anything to do with any variable except i.

pf:

$$W(self, False) \in Assn.$$

$$loc(W(self, False)) = loc(self) = \{x_1, x_{i_2}, \ldots, x_{i_k}\}$$

$\sigma \models^I W(self, false)$ iff $C[\![self]\!]\sigma'$ is undefined for any σ' s.t. $\sigma' =_{loc(self)} \sigma$

W(self, false) says that "self does not halt on the state whose value for the locations in self are precisely $x_1, x_{i_2}, x_{i_3}, \ldots, x_{i_k}$"
$i \quad 0 \quad 0 \quad 0$

this is an assertion about $x_1 \ldots x_{i_k}$.

I want it to talk about when x_1 has value i

So $W(self, false)[i/x_1][0/x_{i_2}]\cdots[0/x_{i_k}]$

is the desired assertion.

Substitution Lemma:

$\sigma[n/x]\models^I A$ iff ~~$\sigma \models^{I[n/i]} A[i/x]$~~ $\sigma \models^{I[n/i]} A[i/x]$

$\sigma \models^I A$ iff $\sigma \models^{I[\sigma(x)/i]} A[i/x]$

Gödel's Incompleteness Thm: The valid Assn's, more precisely, their Gödel numbers, are not Imp-checkable. (Hence, there is no "reasonable", "effective", "checkable" proof system for validity)
an ~~imp~~ IMP pgm which halted on input n iff n was the Gödel number of a valid assertion.

Remark: If we had any reasonable notion of "proof" for valid Assn's we ought to be able to Check* an alleged proof for well-formedness even if we weren't creative enough to discover the proof. (Also, every valid assertion has at least one proof, and if an Assn has a proof, then it is valid)

* by some program which always terminates.

Now if we had such a proof system for valid Assn's, then the valid assertions themselves would be checkable by the following command:

~~$X_1 := 0$; $X_2 := 0$;~~
~~while $X_1 = 0$ do~~

while Flag = 0 do
Checkproof(X_2, X_3); % checks that X_2 is the number of a proof of assertion with number X_3. Sets Flag gets set to 1 or 0, if yes or No.

$X_2 := X_2 + 1$

~~This pgm halts precisely when it finds~~

pf: suppose there was an IMP-checking Valid ∈ Com for Valid Assn's. Then we could get a non-halting checking Not Halt ::= "~~Set X_1 is the Gödel number~~ Construct the Gödel number of the assertion. $S[^n/_i]$ where $n = X_1$, set X_1 to this number, clean up all other registers to zero; do Valid

X. Contradiction

6.044c
p1
11/20

$S \in$ Assn means "com_i does not halt on state $s(i)$".

Let $R(n)$ mean $\models S[^n/_i]$.

So S expresses R

Lemma: R is not IMP-checkable. (By diagonal argument)

~~So S expresses R.~~

Suppose validity of Assn's was checkable: namely, $Valid \in Com$ s.t. $Valid$ halts on $s(n)$ iff $\models A_n$.

A_n ::= the assertion with Gödel number n.

Then R would be IMP checkable, a contradiction.

∴ Theorem: Validity is not IMP-checkable.

Here's the program which would check R:

$X_1 :=$ "Gödel # $(S[^{\sigma(X_2)}/_i])$"; $X_2 :=$ polynomial (X_1)

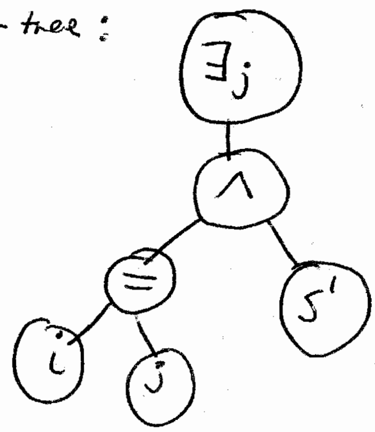
Valid.

WLOG, can assume S is of the form:

$\exists j.(i=j \wedge S')$

no i's in here (↑ S').

parse-tree:



~~Gödel # (S) = mkexist(#j, mkconj(~~

$\#i = 3$

$j \equiv i_2$ ~~$\#j = 2$~~ $i \equiv i_4$

(mkvar 4) (mkvar 2)

$\text{Gödel}\#(S) = \text{mkexist}(\text{(mkvar 2)}, \text{mkconj}(\text{mkeq}(\#i, \#j), \#S'))$

$\text{mkexist}(n, m) = \text{mkpair}(\#\exists, \text{mkpair}(n, m))$

Table	#
+	3
∧	7
∃	11
var	13

$\text{mkpair}(k, l) = 2^k \cdot 3^l$.

(mkvar 2) (mknum(n)) (mkvar(2))

$\text{Gödel}\#(S[n/i]) = \text{mkexist}(\#j, \text{mkconj}(\text{mkeq}(\ \wedge\ , \#j), \#S')$

$2^{11} \cdot 3^{\left(2^{13} \cdot 3^2\right)}$

$= \left(2^{11} \cdot 3^{2^{2^{13} \cdot 3^2} \cdot 3^{2^{\cdots n} \cdot 3^{\cdots}}}\right)$

$\text{Gödel}\#(S[n/i])$

use instead $\text{mkpair}(n,m) = \text{poly}(n,m)$

Logic Speel.

We've seen incompleteness result
we've seen "fake" completeness result.

Now we'll see real completeness result.

Examples of complete axiom systems:
polynomial equations: Peqn ... $e ::= a_1 = a_2$
(for simplicity no integers)

~~Axioms~~ how do I know $x_3 \times (x_1 + x_2) = ((x_1 \times x_3) - 1 + 1) + (x_3 \times x_2)$

$x_1 x_3 + x_2 x_3$

$x_1 \times x_2 \times x_1 = x_2 \times x_3 \times x_1$
$x_1 \times x_3 \times x_2 = x_3 \times x_1 \times x_2$

6.044 p1 11/22

Axiom system for Peqn (arithmetic equations $a_1 = a_2$)

Axiom: $a = a$ (reflexivity)

Rules:

$$\frac{a = a'}{a' = a} \quad \text{(symmetry)}$$

$$\frac{a_1 = a_2,\ a_2 = a_3}{a_1 = a_3} \quad \text{(transitivity)}$$

$$\frac{a_1 = a_2}{a_1 \text{ op } a = a_2 \text{ op } a} \quad \text{(congruence) for op} \equiv +, \times, -$$

$$\frac{a_1 = a_2}{a \text{ op } a_1 = a \text{ op } a_2}$$

<u>Thm:</u> $\models a_1 = a_2$ iff $\vdash a_1 = a_2$.

$\models (x_1 - x_2 \times 3) = 1 \times x_2 \times x_1 - 9 \times x_2 \times x_3$.

<u>pf:</u> ($\Leftarrow$) as usual, because all axioms are $\models$, and rules will preserve validity.

comment on ~~rule~~ what it takes to show $\models a = a$.
comment on what it takes to show (to check defns)
$\models a = a'$ implies $\models a' = a$.

($\Rightarrow$) Idea: use enough rules and axioms so for every a_1 there is a unique canonical form a_2 s.t. $\vdash a_1 = a_2$.

Canonical form will be a sum of monomials with (non-zero) integer distinct coefficients, listed decreasing order of degree, with monomials of same degree listed in "alphabetical" order (x_1 precedes x_2 precedes x_3 ...) written and parenthesized to the left

Then use the fact ~~two canonical forms~~ dist canonical forms have different meanings.

So if $\models a_1 = a_1'$, then $\vdash a_1 = a_2$ ← canonical.

and $\vdash a_1' = a_2'$ ← canonical.

But $\models a_2 = a_2'$

$\therefore a_2 \equiv a_2'$ $\quad \vdash a_1 = a_2$, $\vdash a_1' = a_2$ by sym. $\vdash a_2 = a_1'$

transitivity $\vdash a_2 = a_1'$ Q.E.D

$(3x_1 - 2x_2) + x_3$ ← not a canonical form

$((3 \times x_1) + ((-2) \times x_2)) + (1 \times x_3) \in$ A exp.

$((-1) \times x_2 \times x_2 \times x_2) + (3 \times x_3 \times x_1)$

longer

Axioms: $a = a$

$a_1 + a_2 = a_2 + a_1$

$-a = (-1) \times a$

Distributivity.

$\sim\sim\sim a \times (b + c) \sim\sim\sim$

$\parallel$

$\sim\sim\sim a b + a c \sim\sim\sim$

} this is why we love congruence rule.

derived rule. $\dfrac{a_1 = a_2}{a[a_1/x] = a[a_2/x]}$ by S.I. on a.

G044
pl
11/25

$$\frac{a_1 = a_2}{a_1 + a = a_2 + a}$$

$$\frac{a_1 = a_2}{n + x \sim [a_1] \sim = n + x \sim [a_2] \sim}$$

↑ is a derived rule

$[\;\;] + a$

$a \times (b+c) \overset{\Rightarrow}{=} (a \times b) + (a \times c)$

$(b+c) \times a \Rightarrow (b \times a) + (c \times a)$

$\underline{\times}$

$\underset{\sim}{n} + \underset{\sim}{m} = \underset{\sim}{n+m}$

$0 \times a = 0$
$0 + a = a$

instance $3 + 4 = 7$

$0, 1, 1+1, (1+1)+1,$

Ring

we know how to get poly to NF,
use dist. to flatten
re order
use

a Ring is a Field w/ no inverses.

Today : Axiomatizability and Decidability

IMP-checkable a set of numbers was IMP-checkable if there was a program which halted on precisely that set.

* normal terminology "recursively enumerable"

examples: self-halting set $\{n : com_n$ halts on input $n\}$ is r.e., it's complement is not r.e.

example: Valid assertion of 1^{st} order arithmetic.
not r.e., In fact we showed that the true closed, location free Assn's are not r.e.

Complement of the True closed Assn's is also not r.e.

Pf: Suppose could check that $A \in Assn$ was not closed and True. Then check for True as follows: "Given $B \in Assn$, if B is not closed, then diverge. Otherwise, act like not-True checker on $\neg B$."

Polynomial equality: checkable (r.e.)

Polynomials which are not equal; also checkable.

$$(X_1 X_2^2 + 3 X_2 X_1) X_2 \text{ not } = \text{ to } 4X_1^3 + 7X_2^2 X_1$$

a sum of distinct monomials w/ non zero coeffs if is not syntactically 0, will be 0 sometime

another neat way to check? X_1, X_2, X_3 write a program which cycles through all possible triples of values & checks.

If a set & its complement are checkable then the set is decidable.

Difference between a Decider & Checker.

that is, polynomial equality is decidable.

$$\models \underbrace{3X_1X_2}_{a_1} = \overbrace{X_2X_1 + 2X_1X_2}^{a_2}$$

$$\not\models \underbrace{X_1X_2^2}_{a_3} = \underbrace{X_1^2X_2}_{a_4}$$

a_3 is not equal to a_4.

? $\models X_1X_2^2 \neq X_1^2X_2$ No! these 2 are very different.

$\models 1 \neq 0$ $\qquad \models X_1^2 \neq -1$ $\qquad \models X_1 + 1 \neq X_1$

6.044
p2
11/25

Hilbert's 10th Problem: Valid polynomial inequalities

Find X's st $p(x_1, x_2, \ldots, x_n) = 0$.

+ Procedure should tell you there are none.

so $\models p(x_1, .., x_n) \neq 0$

~~1960~~ 1970 Matijasevic) that Valid polynomial inequalities are __not__ checkable.

$\not\models a_1 \neq a_2$
ie they are equal somewhere!

Shows how to think of polynomials as programming language!

can write a poly that simulates imp, as the values of its args change, all positive values

__Wed.__ Go on with a survey of how R.E. Decidable + Axiomatizable comes up in other subjects.

6.044
p1 Recursive Enumerability & decidability in various
11/27/91 areas:

$\models p(x_1, x_2) \neq q(x_1, x_2)$ — Not checkable (complement is checkable)

$\models p(x_1, x_2) = q(x_1, x_2)$

Sound very similar ~~simple~~, in fact very different

this is an easy problem — this looks almost the same

Decidable — always halts with the answer
iff its complement are IMP-checkable

$p - q \neq 0$

Complement of $\models p(x_1, x_2) \neq q(x_1, x_2)$

is $\{(p, q) : \exists n_1, n_2 .\ p(n_1, n_2) = q(n_1, n_2)\}$
(well, any number of args).

asking if it is possible to
$\models x^2 + y^2 + z^2 \neq 3x^3 - 9y$?

it ~~does~~ not make ~~sense~~ ~~to~~ to ask for a particular question with a yes/no answer whether or not it is decidable.

Logic:

$$\models (\exists x.\ f(x) = x) \Rightarrow (\exists y.\ f(f(y)) = y)$$

1st order predicate calculus.

f : 1 argument

$\not\models \forall x, \exists y\ g(x, y) = g(y, x) \Rightarrow \forall x \forall y.\ g(x, y) = g(y, x)$

consider g as minus

$\models$ $\Leftarrow$

An Interpretation Has more work to do now ...

Thm: Validity for First order predicate calculus is r.e., but not decidable (Gödel + Church)

Word problem for semigroups (1 bin op which is associative)

abb = cb
bbb = c c ⇒ acacb =? bcaca
  ‖
acaabb
NO

goal acacb = bcaca

acacbb
acaabbbb
‖
acaacc

Puzzle: is it possible to write a program which can take a bunch of hypothesized equations.

Thm re — not decidable. Alan Turing.
(re: obvious)

For commutative semigroups it is decidable

add ab = ba
ac = ca
bc = cb.

Fact: Finite State Equivalence. DECIDABLE
Suppose have commutativity of certain inputs ab = ba, cb = bc & ac ≠ ca
Do they accept equivalent strings. Equivalence of FSM is undecidable (under "partial" commutativity)

6.044J/18.423 Semantics of Dataflows

p1 12/2/91

r.e. / not r.e.

decidable / not

be able to recognize from its description that it is obviously r.e. eg set of integer polynomials which have a root

$\models_{Num}$ for ordinary $\models$ over Assn

$\models_{\mathbb{R}}$ interpret variables as ranging over $\mathbb{R}$

Real Assn's

~~Complex Assn's~~

Examples

$\models_{Num} \neg(y \times y = 2)$

$\not\models_{\mathbb{R}} \neg(y \times y = 2)$

$\models_{\mathbb{Q}} \neg(y \times y = 2)$ (latest AMM)

$\models_{\mathbb{C}}$ (w/o $\leq$)

(1) $\models_{Num}$ not r.e. (Gödel's Incompleteness)

(2) $\models_{\mathbb{C}}$ and $\models_{\mathbb{R}}$ are decidable!

1+2 imply that it is not possible to write an Assn w/ fv x, which when interpreted over $\mathbb{R}$ is true iff you interpret x as an integer.

give reduction from (1) to (2) which would therefore exist.

$\widehat{\not\models_{Num} \forall x. F(x)}$ ::= $\models_{\mathbb{R}} \forall x.$ "x is an integer" $\Rightarrow \widehat{F(x)}$

$\widehat{\models_{Num} \exists x. F(x)}$ ::= $\models_{\mathbb{R}} \exists x.$ "x is an integer" $\wedge \widehat{F(x)}$

other case (do nothing)

$\models_{Num} F$ iff $\models_{\mathbb{R}} \hat{F}$

(3) $\models_{\mathbb{Q}}$ is not r.e., there is a number theoretic way to define "x is an integer"!

Add Assn: $\models_{Num}$ same as Assn's but only $+, -$ (no $\times$).
(over nums)
Decidable.

Mult Assn: $\models_{\mathbb{C}}$ only $\times$ (no $+, -,$ or $\leq$)
(over nums)
Decidable

Eq Assn's are Decidable over $\models_{\mathbb{R}}$, $\models_{\mathbb{C}}$, $\models_{\mathbb{Q}}$, $\models_{Num}$
no Arith ops at all.
all we have is variables Nums, ~~no consts~~ coincide
eg $(\forall x. \exists y. x = y \wedge \neg(y = z)) \Rightarrow (\forall w. z = w \vee \neg(z = z))$

in fact over any infinite set it comes out the same.

abbreviate $x < y$.

$\models_{\mathbb{R}}$
$\models_{\mathbb{R}}$
$\not\models_{Num}$

$\forall x. \forall y. (\neg(x = y) \wedge (x \leq y)) \Rightarrow \exists z. x < z \wedge z < y$

6.044L
p1
12/4/91

Problem: Given an imp pgm whether it halts w/ all inputs set to 0.

Thm: It is not decidable whether an ~~imp~~ IMP-program halts when all locations are zero (in state s(0)). That is, $\{n \geq 0 \mid \mathbb{C}om_n \mathbb{I}(s(0)) \neq \bot\}$ ← zero-halt is not ~~[illegible]~~ IMP-decidable.

pf: Strategy: show that if this zero-halting problem was decidable, so would be the self-halting problem, a contradiction.

"a reducibility argument". Show you that if I had a program that was a zero-halt decider then I could get a program that was a self-halt decider.

~~Let~~ For any given $n \geq 0$, let $c_n \in Com$ be "$X_1 := n$; SELF" where self is a self-halting checker.

Now c_n is "input independent", in particular,

~~i c_n halts on state s(0) iff c_n halts on s(k) for all k ∈ Num~~
~~iff c_n halts on state s(#c_n)~~
~~iff A [illegible] self-halting~~
~~iff #c_n~~

c_n halts on state s(0) iff ~~self~~ SELF halts on s(n)
$\#c_n$ = ~~mkseq~~ mkseq(mkassign(mkvar(1), mknum(n)), #self) iff ~~com_n halts on s(n)~~
n is in the self-halting set.

Thus the "IMP program described in imp meta-language as follows is a self halting decider

$X_1 = \#c_{X_1}$;
ZERO-DECIDER

} would be a self-halting decider

✱

Exercise: This same argument shows that if ZERO-NON-HALTING were checkable, so would be SELF-NON-halting.

reducibility

Simulation and minimal "RISC" machines.

A "counter machine":

locations	$x_1, x_2, \ldots$	
assignment	$INC(x_i)$	$(x_i := x_i + 1)$
	$DEC(x_i)$	$(x_i := x_i - 1)$
test	$BGZ(x_i, l_1, l_2)$ ← labels	(if $x_i \geq 0$ goto l_1 else goto l_2)

Program: a sequence of statements (one line commands) each with a distinct label.
start at top
branch to label that's not there — halt.

Let's write a counter machine program for "$x_2 := x_2 \bmod 3$"
assume $\sigma(x_i) = 0$ for $i \geq 2$.

~~copy~~ "destructively copy x_1 into x_3, x_4." assume $x_1 \geq 0$.

l_0: ~~BGZ(x_1,~~ $DEC(x_1)$
l_1: $BGZ(x_1,$ ~~halt~~ l_4, $l_2)$
l_2: $INC(x_2)$
l_3: $BGZ(x_2, halt, halt)$
l_4: $INC(x_3)$
l_5: $INC(x_4)$
l_6: $BGZ(x_1, l_0,$ ~~halt~~ $l_0)$

6.044
p1
12/6/91

Counter machines:

INC(X)
DEC(X)
BGZ(X, l_1, l_2)

The zero-halting problem for counter machines is undecidable.

How to compile imp code to risc machine.

"$x_3 := x_2$;
$x_4 := x_2$;
$x_2 := 0$"

Likewise "$x_1 := x_2$"

Do this by checking that $x_2 \geq 0$, and if so decrementing x_2 while incrementing each of x_3 and x_4 until $x_2 < 0$. The other case, when $x_2 < 0$ is the same with increment & decrement and the ≤ test reversed.

We'll see the halting problem for machine w/ 2 counters is undecidable.

Building up counter machines w/ macros, & compiling down Imp code.

"$X_2 := X_1 \bmod 3$"

dec x_1 3 times until x_1 goes negative.
inc 3 times after that you had to reverse

$x_2 := \lfloor X/3 \rfloor$

To "simulate" 5 counters with 2:

State $\langle n_1, n_2, n_3, n_4, n_5 \rangle$ gets represented

$2^{n_1} \cdot 3^{n_2} \cdot 5^{n_3} \cdot 7^{n_4} \cdot 11^{n_5}$ ← this particular representation of pairing matters

really $\langle 2^{sg(n_1)} \cdot 3^{|n_1|} \cdot 5^{sg(n_2)}, 7^{|n_2|} \cdot 11^{sg(n_3)} \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29^{|n_5|}, 0 \rangle$

$sg(n_1) = 1$ if $n_1 \geq 0$
$= 0$ if $n_1 < 0$

~~suppose m 5 c~~
5 ctr machine

determine.

INC(x_2)

X_1: if $sg(x_2)=1$ 7 x X_1,
else $\lfloor X_2 / 7 \rfloor$ ...

BGZ(x_3, l_1, l_2)

branch on X_1 mod 11

Imp instructions.

$X_1 := x_2 + x_3$ assume $x_2, x_3 \geq 0$ upon which set X_1 to 0.
dec x_2 while inc'ing x_1
then dec x_3 while inc'ing x_1

$X_1 := x_2 \times X_3$

Compiling IMP code.

Simple Com ::= simple-ass | simple-ass; simple com

Simple ass ::= $X := n$ | $X := X' \, op \, X''$.

$a \in Aexp$

$\hat{a}(Z)$ ~~A~~ $\in$ ~~Com~~ Simple Comm $Z \in Loc - loc(a)$

$[\![\hat{a}(Z)]\!] = [\![Z := a]\!]$

$\hat{a}(Z)$ sim's up to temps $Z := a$

$\hat{n}(Z)$ is $Z := n$ ← temps (initially 0)

$\hat{X}(Z)$ is ~~X := ... + T~~ $Z := X + T$

$\widehat{a_1 + a_2}(Z)$ is $\hat{a}_1(T_1)$; $\hat{a}_2(T_2)$; ~~x~~ $Z := T_1 + T_2$.

← T_1 is fresh temp ← a fresh one

if $(a_1 \geq a_2)$ then c_1 else c_2.

~~b_1~~ convert this to $X \geq 0$.

$T := a_1 - a_2$
if $T \geq 0$ then c_1 else c_2.

$b_1 \wedge b_2$.

6044 pst
12/6/91

if b then c_1 else c_2

$\hat{b}(T)$; if $T \geq 0$ then $\hat{c_1}$ else $\hat{c_2}$

SEND E-Mail Quiz Review
Recap wed.

6.044C
p1
12/9

Milne & Strachey
→ semantics
ALGOL 60.

"Applied Theory"

Where we didn't get a chance to go.

Aexp's with $\underset{\sim}{f}(\ ,\ )$ function symbols.

$m_0 \to m'$ & $m \le N$ then either $m' \le N$
or $\exists N'$ $N \to N'$
& $m' \le N'$

$a ::= f(a_1, a_2) \mid \ldots \mid$ if $a_1 = 0$ then a_2 else a_3.

terms

$F(t_3) ::=$ let $f_1(x_1, t_2)$ be t_1 and $f_2(x_1)$ be t_2 in t_3

$$\frac{a_1 \to n_1 \qquad a_2 \to n_2}{a_1 + a_2 \to \underline{n_1 + n_2}}$$

~~$F(a_1 + a_2)$~~

$$\frac{F(a_1) \to n_1 \qquad F(a_2) = n_2}{F(a_1 + a_2) \to \underline{n_1 + n_2}}$$

Call-by-name

$$\frac{F(t_2[a_1/x_1, a_2/x_2]) \to n}{F(f_2(a_1, a_2)) \to n}$$

Bekic's Thm: deepest declaration

let $f_1(x_1, x_2)$ be

t_1 — $t_1[D/f_2]$

in let $f_2(x_1)$ be

~~t~~ t_2

in

t_3

D ::= let $f_2(x_2)$ be t_2

t_3 ::= $f_2(3) + f_2(f_2(a))$

$(\text{let } f_2(x_2) \text{ be } t_2 \text{ in } f_2(3)) + \text{let } f_2(x_1) \text{ be } t_2 \text{ in } f_2(f_2(a)))$

sketch proof using level fixpoint reasoning.

6.044 p1 12/11/91

Handout 12pp. on "Notes on Decidability of Checkability"

Quiz 4
Checkable_re
Decidable
Expressible

Decidable → Checkable

pf: Suppose c is a decider for something how can we turn it into a checker

prove that the intersection of decidable sets is decidable.

Lemma: $M_1, M_2 \subseteq Num$, decidable, then $M_1 \cap M_2$ is decidable.

Pf: Say c_1 decides M_1
c_2 decides M_2

c_3 will decide $M_1 \cap M_2$;

$c_3 ::= c_1$; $T := x_1$; if $x_1 = 1$ then $x_1 := T$; c_2 else skip;; $T := 0$

buggy

This very program also works if c_1 & c_2 are only checkers
checkable sets are also closed under union, but designing such a one is harder

Actual problems: Truth
Self-halting
Polynomials
"easy fragments of Assn"

extend ~~polynomials~~ Aexps to be "conditional-polynomials"

if $a_1 \geq 0$ then a_2 else a_3

cond Aexps.

Practice Problem.

is it decidable if $\models a_1 = a_2$, for $a_1, a_2 \in$ Cond Aexp's.
checkable?
is the complement checkable?

Zero-state halting problem.
Gödel numbering.

Counter Machines.
Universal Simulator
INDUCTIVE Score Checkable.

Follow up class

6.830J / 18.427 6.821

6.826? Lampson & Weihl
Prof. Guttag EECS. Larch Prover

Hard to read text old.

Course Information

Staff.

Lecturer:	Prof. Albert R. Meyer `meyer@theory.lcs.mit.edu`	NE43-315	x3-6024
Teaching Assistant:	Arthur F. Lent `aflent@theory.lcs.mit.edu`	NE43-344	x3-6259
Secretary:	David Jones `6044-secretary@theory.lcs.mit.edu`	NE43-316	x3-5936

Lectures and Tutorials. Class meets MWF from 1:00–2:00 PM in 2-146. There will be no recitation sections, but tutorial/review sessions may be organized in response to requests. The TA will have one regularly scheduled office hour to be announced the first day of class. Further meetings with the TA or instructor can be scheduled by appointment.

Prerequisites. The official requirement for the course is either 18.063 *Introduction to Algebraic Systems*, or 18.310 *Principles of Applied Mathematics.* If you know the basic vocabulary of mathematics and how to do elementary proofs, then you may take this course with the permission of the instructor.

Contrarequisites. There will be less overlap with 6.045J/18.400J and this course than in previous terms, so Course 6 students gung-ho for theory will no longer be discouraged from taking both courses. There will be a smaller overlap with 6.840J/18.404J; students, especially Math majors, *may* routinely take both this course and 6.840J/18.404J.

Textbook. The required text for the course is *Introduction to the Formal Semantics of Programming Languages* by Glynn Winskel. The book is in manuscript form and will be xeroxed and handed out in class. Students will be minimally charged for reproduction costs.

Grading. There will be regular problem sets, quizzes, and most likely a regular three hour final exam. This will be decided in class the at the beginning of the term. The problem sets, quizzes, and final each *count about equally* toward the final grade.

Problem Sets. There will be likely be six to eight problem sets. Homework will usually be assigned on a Friday and due 7–10 days later.

Handouts and Notebook. You may find it useful to get a loose-leaf notebook for use with the course, since all handouts and homework will be on standard three-hole punched paper. If you fail to obtain a handout in lecture, you can get a copy from the file cabinet to the right of the door to room NE43-311. If you take the last copy of a handout, please inform David in room NE43-316 so that more copies can be made.

Handouts will also be available via anonymous ftp from **theory.lcs.mit.edu**. To retreive these files, run **ftp**, and open **theory.lcs.mit.edu**, supplying "anonymous" as the name (account) and "guest" as the password. All handouts are written in LaTeX and will be placed in the directory "pub/6044". Files may then be retrieved by first typing "**cd pub/6044**" to change directories and then typing "**get** *filename*." You will need the files **6044.sty** and **handouts-6044-fall-91.tex** (which serves as an index to the handouts) in order to run LaTeX on the handout files.

The handouts can also be retrieved via mail server. For more information, send a message to **archive-server@theory.lcs.mit.edu** with the single word "help" in the body.

Electronic mail. All students are encouraged to subscribe to the course mail list by sending email to **6044-secretary@theory.lcs.mit.edu**; other administrative requests should also be directed to this address.

To facilitate communication in the class, there are three electronic mail addresses:

6044-secretary@theory.lcs.mit.edu
6044-forum@theory.lcs.mit.edu
6044-staff@theory.lcs.mit.edu

The **6044-forum** mailing list is for general communication by students, the instructor, and the TA to the class; a message sent here will automatically be distributed to those on the mailing list. Students are strongly encouraged to use **6044-forum** to arrange study sessions, discuss ambiguities and problems with homework, and send comments to the whole class. The TA and instructor may also post bugs and corrections to homeworks and handouts to **6044-forum**.

Messages to the instructor, TA, or grader should be sent to **6044-staff**.

Pictures. You can help us learn who you are by giving us your photograph with your name on it. This is especially helpful if you later need a recommendation.

Diagnostic Quiz

You will not be graded on this quiz. Do not discuss it with anyone before taking it. Take it sometime after class, and return it to the TA on Friday, September 13. Be sure to indicate your name, the date, "6.044 Diagnostic Quiz", and the time it took you, on your answer sheet.

Problem 1. Describe the function which is the composition of the integer successor function, *i.e.*, successor$(x) = x + 1$, with itself.

Problem 2. How many strings of length four are there over the alphabet $\{a, b, c\}$?

Problem 3. Give an example of an uncountable set.

Problem 4. Which is a synonym for "injective"?

(a) epi

(b) onto

(c) mono

(d) isomorphism

(e) one-to-one

(f) one-to-one and onto

What sets have the property that there is *no* injection from the set into itself?

What sets have the property that there is *no* injection from the set into a *proper* subset of itself?

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \rightarrow B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a countererxample.

(a) reflexive

(b) symmetric

(c) transitive

(d) equivalence relation

(e) partial order

Problem 6. Describe a propositional, *i.e.*, Boolean, connective which is not commutative.

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are ***equivalent*** iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables.

(b) Explain why "equivalence" is actually an equivalence relation on formulas.

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many?

Solutions to Diagnostic Quiz

Problem 1. Describe the function which is the composition of the integer successor function, *i.e.*, successor$(x) = x + 1$, with itself. Answer: $x + 2$.

Problem 2. How many strings of length four are there over the alphabet $\{a, b, c\}$? Answer: $3 * 3 * 3 * 3 = 81$; for each position there are three possible letters, and there are 4 possible positions.

Problem 3. Give an example of an uncountable set. Examples: the real numbers, and the real numbers between 0 and 1.

Problem 4. Which is a synonym for "injective"? Answer: **(e)** one-to-one.

What sets have the property that there is *no* injection from the set into itself? Answer: NONE. The identify function from a set onto itself is always well-defined, and always an injection.

What sets have the property that there is *no* injection from the set into a *proper* subset of itself? Answer: Precisely the finite sets.

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \rightarrow B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a counterexample.

- **(a)** reflexive. Answer: YES. The identity from A to A always exists and is always injective.
- **(b)** symmetric. Answer: NO. Consider $A = \{1\}$ and $B = \{1, 2\}$. $A \preceq B$ but $B \not\preceq A$.
- **(c)** transitive. Answer: YES. If f_1 is an injection from A to B and f_2 is an injection from B to C then $f_2 \circ f_1$ is an injection from A to C.
- **(d)** equivalence relation. Answer: NO. A relation is an equivalence relation iff it is reflexive, symmetric and transitive. $\preceq$ is not symmetric.
- **(e)** partial order. Answer: No. A relation is a partial order iff it is reflexive, transitive, and ***anti-symmetric***, *i.e.*, if A is related to B and B is related to A then $A = B$. If we consider the case of $A = \{1, 2\}$ and $B = \{3, 4\}$, then $A \preceq B$ and $B \preceq A$, but $A \neq B$.

Problem 6. Describe a propositional, *i.e.*, Boolean, connective which is not commutative. Answer: Implies ($\supset$) is a propositional connective which is not commutative. (8 of the 16 propositional connectives are not commutative).

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are *equivalent* iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables. Example: $x_1 \supset x_2$, $\overline{x_1} \vee x_2$ and $\overline{x_1} \vee x_2 \vee x_2$ are true for all assignments *except* $x_1 = \mathtt{true}$ and $x_2 = \mathtt{false}$, in which case all are false.

(b) Explain why "equivalence" is actually an equivalence relation on formulas. Answer: Because it is reflexive (obviously), symmetric (if F_2 agrees with F_1 on all input values, then the opposite must also be the case), and transitive (if F_1 agrees with F_2 on all inputs values, and F_2 agrees with F_3 on all input values, then F_1 agrees with F_3 on all input values), by definition the relation "equivalence" is an equivalence relation on formulas.

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many? Answer: For n variables there are exactly 2^n different 0-1 assignments to the variables. For each assignment to the variables there are two possible truth values to yield. Consequently there can be at most only 2^{2^n} different equivalence classes. Why? By the pigeonhole principle if there were more than this 2^{2^n} equivalence classes then at least two of them would have to have the same input/output behavior, in which case they would be the same equivalence classes, so there can be at most 2^{2^n} distinct equivalence classes.

Instructions for Problem Sets

1 Form of Solutions

Each problem is to be done on a *separate sheet* of *three-hole punched paper*. If a problem requires more than one sheet, staple these sheets together, but keep each problem separate. Do not use red ink. Mark the top of the paper with:

- Your name,
- "6.044J/18.423J",
- the assignment number,
- the problem number, and
- the date.

Try to be as clear and precise as possible in your presentations. Problem grades are based not only on getting the right answer or otherwise demonstrating that you understand how a solution goes, but also on your ability to explain the solution or proof in a way helpful to a reader.

If you have doubts about the way your homework has been graded, first see the TA. Other questions and suggestions will be welcomed by both the instructor and the TA.

Problem sets will be collected at the beginning of class; graded problem sets will be returned at the end of class. Solutions will generally be available with the graded problem sets, one week after their submission.

2 Collaboration and References

You must write your own problem solutions and other assigned course work in your own words and entirely alone. On the other hand, you are encouraged to discuss the problems with one or two classmates before you write your solutions. If you do so, please be sure to

indicate the members of your discussion group

on your solution.

Similarly, you are welcome to use other texts and references in doing homework, but if you find that a solution to an assigned problem has been given in such a reference, you should nevertheless rewrite the solution in your own words and *cite your source*.

3 Late Policy

Late homeworks should be submitted to the TA. If they can be graded without inconvenience, they will be. Late homeworks that are not graded will be kept for reference until after the final. No homework will be accepted after the solutions have been given out.

Problem Set 1

Due: 20 September 1991.

Remark. Before beginning the assignment, please be sure to read Handout 4, "Instructions for Problem Sets."

Problem 1. Write down a full set of rules for $\rightarrow_1$ on command configurations, so $\rightarrow_1$ stands for a single step in the execution of a command from a particular state, as discussed on page 25 of the text. Use command configurations of the form $\langle c, \sigma \rangle$ and σ when there is no more command left to execute. Point out where you have made a choice in the rules between alternative understandings of what constitutes a single step in the execution.

(Showing $\langle c, \sigma \rangle \rightarrow_1^* \sigma'$ iff $\langle c, \sigma \rangle \rightarrow \sigma'$ is hard and requires the application of induction principles introduced in chapters 3 and 4—you are not expected to show this now).

Hint: The rule for **while** should be:

$$\langle \textbf{while } b \textbf{ do } c, \sigma \rangle \rightarrow_1 \langle \textbf{if } b \textbf{ then } (c; \textbf{while } b \textbf{ do } c) \textbf{ else skip}, \sigma \rangle$$

Problem 2. In our language, the evaluation of expressions has no side effects—their evaluation does not change the state. If we were to model side-effects, it would be natural to consider instead an evaluation relation of the form

$$\langle a, \sigma \rangle \rightarrow \langle n, \sigma' \rangle$$

where σ' is the state that results from the evaluation of a in original state σ. To introduce side effects into the evaluation of arithmetic expressions of **IMP**, extend them by adding a construct

$$c \textbf{ resultis a}$$

where c is a command and a is an arithmetic expression. To evaluate such an expression, the command c is first executed and then a evaluated in the changed state. Formalize this idea by first giving the full syntax of the language and then giving it an operational semantics.

Problem Set 2

Due: 27 September 1991.

Problem 2 is rather long, so start early.

Reading assignment. Winskel through Section 4.3.

Problem 1. [Deterministic Rewriting] Let γ denote a command configuration of the while programming language **IMP**, and δ denote either a command configuration or a state.

1(a). Prove by structural induction that for every γ, there is exactly one δ such that $\gamma \rightarrow_1 \delta$. Briefly comment on where structural induction would break down for a similar proof about the "evaluates to" relation.

1(b). Conclude that there is a partial function

$$eval : [\text{command configurations}] \rightarrow \Sigma$$

such that for all states $\sigma \in \Sigma$

$$\gamma \rightarrow_1^* \sigma \text{ iff } eval(\gamma) = \sigma.$$

Problem 2. This problem is based on the exercise in Winskel, p. 47, proving equivalence of rewriting and evaluation semantics of **IMP** commands. We have broken the problem up into several "independent subproblems." If you are unable to do a part, go on to the next part. For all later parts, you may assume that you have done all of the other other parts correctly.

Handout 8 given out on September 20, and handout 9 which will be given out on September 23, will form our official definition of the relation $\rightarrow_1$ (the one step execution relation). The goal of this problem is to prove the following Theorem:

$$\forall \sigma, \sigma' [\langle c, \sigma \rangle \rightarrow_1^* \sigma' \text{ iff } \langle c, \sigma \rangle \rightarrow \sigma'].$$

For this problem, you may assume that for $a \in \textbf{Aexp}$, $n \in Num$, $b \in \textbf{Bexp}, t \in \{\texttt{true}, \texttt{false}\}$:

$$\langle a, \sigma \rangle \rightarrow_1^* \langle n, \sigma \rangle \text{ iff } \langle a, \sigma \rangle \rightarrow n$$

and

$$\langle b, \sigma \rangle \rightarrow_1^* \langle b, \sigma \rangle \text{ iff } \langle b, \sigma \rangle \rightarrow b$$

2(a). Prove

$$\forall\sigma,\sigma'.[\langle c_0;c_1,\sigma\rangle \rightarrow_1^* \sigma' \Rightarrow \exists\sigma''.\langle c_0,\sigma\rangle \rightarrow_1^* \sigma''\&\langle c_1,\sigma''\rangle \rightarrow_1^* \sigma']$$

by induction on the definition of $\rightarrow_1^*$ as the ***transitive closure*** of $\rightarrow_1$, in the execution of $\langle c_0;c_1,\sigma\rangle \rightarrow_1^* \sigma'$.

2(b). Prove

$$\forall\sigma,\sigma',\sigma''.[\langle c_0,\sigma\rangle \rightarrow_1^* \sigma''\&\langle c_1,\sigma''\rangle \rightarrow_1^* \sigma' \Rightarrow \langle c_0;c_1,\sigma\rangle \rightarrow_1^* \sigma']$$

again by induction on the definition of $\rightarrow_1^*$ as the transitive closure of $\rightarrow_1$, this time using the execution of $\langle c_0,\sigma\rangle \rightarrow_1^* \sigma''$.

From parts (a) and (b) we can then conclude the Lemma:

$$\langle c_0;c_1,\sigma\rangle \rightarrow_1^* \sigma' \text{ iff } \exists\sigma''.\langle c_0,\sigma\rangle \rightarrow_1^* \sigma''\&\langle c_1,\sigma''\rangle \rightarrow_1^* \sigma',$$

for all commands c_0, c_1, and states σ, σ'. This Lemma will be essential to finish proving the Theorem.

2(c). Prove

$$\forall\sigma,\sigma'.[\langle c,\sigma\rangle \rightarrow_1^* \sigma' \Rightarrow \langle c,\sigma\rangle \rightarrow \sigma']$$

by structural induction on c. Do all cases except for that of c being a **while** loop.

2(d). Do the case of c a **while** loop for the proof of part (c). This can be proven by induction on the length of the computation.

2(e). Prove

$$\forall\sigma,\sigma'.[\langle c,\sigma\rangle \rightarrow \sigma' \Rightarrow \langle c,\sigma\rangle \rightarrow_1^* \sigma']$$

by rule induction (or by induction on derivations).

Parts (c), (d), and (e) give us our main goal.

Preliminary Quiz Schedule

	Date	Time	Room
Quiz 1	Monday, October 7, 1991	1:00-2:00	34-302
Quiz 2	Monday, October 28, 1991	1:00-2:00	34-302
Quiz 3	SEE BELOW		
Quiz 4	During Exam Week		

Quiz 3 will take place either in class on Monday, November 18, or, subject to unanimous class agreement, in the evening on either Tuesday or Wednesday, November 19 or 20.

Class selected Tuesday 7-9pm.

$\rightarrow_1$ Rules for Aexp and Bexp

1 Rules for Aexp's

$$\langle X, \sigma \rangle \rightarrow_1 \langle \sigma(X), \sigma \rangle$$

In the following rules **op** ranges over syntactic symbols, and *op* ranges over the actual functions, as indicated by the chart following the rules.

$$\frac{\langle a_0, \sigma \rangle \rightarrow_1 \langle a_0', \sigma \rangle}{\langle a_0 \ \mathbf{op} \ a_1, \sigma \rangle \rightarrow_1 \langle a_0' \ \mathbf{op} \ a_1, \sigma \rangle}$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow_1 \langle a_1', \sigma \rangle}{\langle n \ \mathbf{op} \ a_1, \sigma \rangle \rightarrow_1 \langle n \ \mathbf{op} \ a_1', \sigma \rangle}$$

$$\langle n \ \mathbf{op} \ m, \sigma \rangle \rightarrow_1 \langle n \ op \ m, \sigma \rangle$$

op	*op*
+	the sum function
−	the subtraction function
×	the multiplication function

Notice that

$$\langle 5 + 7, \sigma \rangle \rightarrow_1 \langle 12, \sigma \rangle$$

is an instance of the rule $\langle n \ \mathbf{op} \ m, \sigma \rangle \rightarrow_1 \langle n \ op \ m, \sigma \rangle$, but that

$$\langle 5 + 7, \sigma \rangle \rightarrow_1 \langle 5 + 7, \sigma \rangle$$

is **NOT** derivable at all.

2 Rules for Bexp's

The following are the general rules for the relational operators. Again, **op** ranges over syntactic objects, and *op* over the actual mathematical functions as indicated in the chart following the rules.

$$\frac{\langle a_0, \sigma\rangle \rightarrow_1 \langle a_0', \sigma\rangle}{\langle a_0 \ \mathbf{op} \ a_1, \sigma\rangle \rightarrow_1 \langle a_0' \ \mathbf{op} \ a_1, \sigma\rangle}$$

$$\frac{\langle a_1, \sigma\rangle \rightarrow_1 \langle a_1', \sigma\rangle}{\langle n \ \mathbf{op} \ a_1, \sigma\rangle \rightarrow_1 \langle n \ \mathbf{op} \ a_1', \sigma\rangle}$$

$$\langle n \ \mathbf{op} \ m, \sigma\rangle \rightarrow_1 \langle n \ op \ m, \sigma\rangle$$

op	*op*
$=$	the equality test function
$\leq$	the less than or equal to function

Once again, notice that

$$\langle 7 \leq 5, \sigma\rangle \rightarrow_1 \langle \mathtt{false}, \sigma\rangle$$

is an instance of the rule $\langle n \ \mathbf{op} \ m, \sigma\rangle \rightarrow_1 \langle n \ op \ m, \sigma\rangle$ whereas

$$\langle 7 \leq 5, \sigma\rangle \rightarrow_1 \langle 7 \leq 5, \sigma\rangle$$

is not derivable at all.

We next have the rules for boolean negation.

$$\frac{\langle b, \sigma\rangle \rightarrow_1 \langle b', \sigma\rangle}{\langle \neg b, \sigma\rangle \rightarrow_1 \langle \neg b', \sigma\rangle}$$

$$\langle \neg\mathtt{true}, \sigma\rangle \rightarrow_1 \langle \mathtt{false}, \sigma\rangle$$
$$\langle \neg\mathtt{false}, \sigma\rangle \rightarrow_1 \langle \mathtt{true}, \sigma\rangle$$

Finally we have the rules for binary boolean operators. We use **op** and *op* to range over the symbols and functions in the chart following the rules.

$$\frac{\langle b_0, \sigma\rangle \rightarrow_1 \langle b_0', \sigma\rangle}{\langle b_0 \ \mathbf{op} \ b_1, \sigma\rangle \rightarrow_1 \langle b_0' \ \mathbf{op} \ b_1, \sigma\rangle}$$

$$\frac{\langle b_1, \sigma\rangle \rightarrow_1 \langle b_1', \sigma\rangle}{\langle n \ \mathbf{op} \ b_1, \sigma\rangle \rightarrow_1 \langle n \ \mathbf{op} \ b_1', \sigma\rangle}$$

$$\langle n \ \mathbf{op} \ m, \sigma\rangle \rightarrow_1 \langle n \ op \ m, \sigma\rangle$$

op	*op*
$\wedge$	the conjunction function (boolean AND)
$\vee$	the disjunction function (boolean OR)

$\rightarrow_1$ Rules for Com

Atomic Commands:

$$\langle \mathbf{skip}, \sigma \rangle \rightarrow_1 \sigma$$

$$\frac{\langle a, \sigma \rangle \rightarrow_1 \langle a', \sigma \rangle}{\langle X := a, \sigma \rangle \rightarrow_1 \langle X := a', \sigma \rangle}$$

$$\langle X := n, \sigma \rangle \rightarrow_1 \sigma[n/X]$$

Sequencing:

$$\frac{\langle c_0, \sigma \rangle \rightarrow_1 \langle c_0', \sigma' \rangle}{\langle c_0; c_1, \sigma \rangle \rightarrow_1 \langle c_0'; c_1, \sigma' \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \rightarrow_1 \sigma'}{\langle c_0; c_1, \sigma \rangle \rightarrow_1 \langle c_1, \sigma' \rangle}$$

Conditionals:

$$\frac{\langle b, \sigma \rangle \rightarrow_1 \langle b', \sigma \rangle}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma \rangle \rightarrow_1 \langle \mathbf{if}\, b'\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma \rangle}$$

$$\langle \mathbf{if\, true\, then}\, c_0\, \mathbf{else}\, c_1, \sigma \rangle \rightarrow_1 \langle c_0, \sigma \rangle$$

$$\langle \mathbf{if\, false\, then}\, c_0\, \mathbf{else}\, c_1, \sigma \rangle \rightarrow_1 \langle c_1, \sigma \rangle$$

While-loops:

$$\langle \mathbf{while}\, b\, \mathbf{do}\, c, \sigma \rangle \rightarrow_1 \langle \mathbf{if}\, b\, \mathbf{then}(c; \mathbf{while}\, b\, \mathbf{do}\, c)\, \mathbf{else\, skip}, \sigma \rangle$$

Lecture Outline: 1–12

1. (9/11) buzzwords: logic and semantics of programs, basic ideas of logic–incompleteness and completeness–and undecidability. Initial study: while-programming language **Imp**. Syntax of **Imp**.

2. (9/13) evaluation "natural semantics" of **Imp**. States $\Sigma = \text{Loc} \to \text{Num}$. Evaluation triples, (a,state,integer) for **Aexp** written $\langle a, \text{state}\rangle \to \text{integer}, \ldots$, (c,state,state) for **Com**. Derivation tree for $\langle (M+N) \times N, \sigma[10/N][6/M]\rangle \to 160$.

3. (9/16) *Configurations* $\langle a, \sigma\rangle$ for **Aexp**, $\langle c, \sigma\rangle$ for **Com**. Derivation tree for

$$\langle \text{Euclid}, \sigma[10/\text{N}][6/\text{M}]\rangle \to \sigma[2/M][2/N]$$

(Euclid on p.32). Mention uniqueness of derivation tree for each configuration; always exists for **Aexp**, **Bexp**, and *while-free* **Com**. Maybe none for general **Com**. Direct algorithm for $\langle c, \sigma\rangle$ of complexity "linear" in size of derivation. Local checkability of derivation trees, even for enriched systems w/o unique derivations.

Thm: $\langle a, \sigma\rangle$ evaluates to *exactly* one n; likewise $\langle b, \sigma\rangle$ evals to exactly one $\nu \in \{\texttt{true}, \texttt{false}\}$. And $\langle c, \sigma\rangle$ evals to *at most* one σ'. No proof.

(from Exercise, p.21): Prove by minimum principle on derivations that $\langle \textbf{while true do}\, c, \sigma\rangle \not\to$ for all c, σ.

4. (9/18) (by aflent). Def of rewriting semantics for **Aexp** (p.24).

Lemma: $\langle a, \sigma\rangle \to_1^* \langle n, \sigma\rangle$ iff $\langle a, \sigma\rangle \to n$. Also $\langle c, \sigma\rangle \to_1^* \sigma'$ iff $\langle c, \sigma\rangle \to \sigma'$. Pfs postponed.

Def. of command equivalence:

$$c_1 \sim c_2 \text{ iff } [\forall \sigma, \sigma'.\ \langle c_1, \sigma\rangle \to \sigma' \text{ iff } \langle c_2, \sigma\rangle \to \sigma'].$$

Lemma (p.22 Prop.1): $\textbf{while}\, b\, \textbf{do}\, c \sim \textbf{if}\, b\, \textbf{then}\, c; w\, \textbf{else}\, \textbf{skip}$.

Proof by cases on the *form* of evaluation derivations.

5. 9/20. Complete proof of above lemma p.22. Induction principle: integer inductions; structural induction. Prove: functionality of evals-to on **Aexp**(Prop.2, p.29). Comment: functionality also holds for commands, but not by structural induction because of **while**. Start on proof of: $\langle a, \sigma\rangle \to_1^* \langle n, \sigma\rangle$ iff $\langle a, \sigma\rangle \to n$.

6. 9/23 (by wald) continue proof of above lemma, discussing structural induction and induction on length of rewriting chain. Prove functionality of command evaluation by induction on derivations:

Lemma: $\langle c, \sigma\rangle \to \sigma'$ and $\langle c, \sigma\rangle \to \sigma''$ implies $\sigma' = \sigma''$.

Rule induction: if a property is preserved by all the rules (including the premise-free rules, eg, the axioms) which define a set inductively, then it is true of all the elements in the set.

Comment: induction on derivations is indistinguishable from Winskel's "rule induction" when derivations are unique, and we won't be picky about the distinction.

7. 9/25 (by wald) well-founded induction, minimum principle p.31. Product of well-founded sets is well-founded under pairwise partial order. (Mention lexicographic p.o. on pairs?) Proof that Euclid terminates on positive inputs (p.33, Thm.4.)

8. 9/27 (by wald) Function def's by induction. Def of $loc_L(c)$, p.37 and proof using it: Prop.8 on p.45.

   Def by structural induction ok on "free" syntactic structures, not for:

   $$f(n+m) = \begin{cases} 1 & \text{if } n+m \leq 1, \\ f(n) + 2f(m) & \text{otherwise.} \end{cases}$$

9. 9/30 Denotational semantics of **IMP**, defined by structural induction. Start on least fixed points.

10. 10/2 More on least fixed points (Winskel §4.4). Equiv of operational and denotational semantics of **IMP** (Thm.15, p.59).

11. 10/4 Complete proof of Thm.15. Brief hint about cpo's (§5.4).

12. 10/7 Quiz 1: Winskel through §5.2 and statement, but not proof, of equiv of operational and denotational semantics of **IMP** (§5.3, p.59, Thm.15).

Equivalence between $\rightarrow_1^*$ and $\rightarrow$ for Aexps

Theorem 1. For all $a \in \textbf{Aexp}$, $n \in \textbf{Num}$, $\sigma \in \Sigma$:

$$\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle \text{ iff } \langle a, \sigma\rangle \rightarrow n$$

1 Proving the "only if" direction

We first prove the "only if" direction ($\Rightarrow$). So, we must show that for an arbitrary $a \in \textbf{Aexp}$, an arbitrary $n \in \textbf{Num}$, and an arbitrary $\sigma \in \Sigma$, if it is the case that $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$, then it is also the case that $\langle a, \sigma\rangle \rightarrow n$.

We now prove the following Lemma which captures most of the complexity of this direction:

Lemma 1. If $\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle$ and $\langle a', \sigma\rangle \rightarrow n$, then $\langle a, \sigma\rangle \rightarrow n$

Proof: The proof of this Lemma is by induction on the structure of a.

Base Cases: The base cases for an induction on the structure of **Aexp**'s are Numerals and Locations.

$a \equiv m$: This case is vacuous since $\langle a, \sigma\rangle$ does not rewrite to anything (*viz.* $\langle a, \sigma\rangle \not\rightarrow_1$).

$a \equiv X$: For this case, the definition of $\rightarrow_1$ gives $\langle a, \sigma\rangle \rightarrow_1 \langle \sigma(X), \sigma\rangle$. Since the supposition of the Lemma is that $\langle \sigma(X), \sigma\rangle \rightarrow n$, then by the definition of $\rightarrow$, it must be the case that $\sigma(X) = n$. As $\langle a, \sigma\rangle \rightarrow \sigma(x)$ (by definition of $\rightarrow$), and $\sigma(X) = n$, we have $\langle a, \sigma\rangle \rightarrow n$.

Inductive Cases: The inductive cases are all those **Aexp**'s which are of the form $a_1 + a_2$, $a_1 \times a_2$, or $a_1 - a_2$. Notice that that a_1 and a_2 range over the full set of **Aexp**'s—they themselves can be numbers, locations, or compound arithmetic expressions. We work out the case of $a \equiv a_1 + a_2$ in detail here. The cases of $a \equiv a_1 \times a_2$ and $a \equiv a_1 - a_2$ follow similarly, with trivial modifications to account for the change in arithmetic operator.

We have several cases. One for each $\rightarrow_1$-rule according to how $\langle a, \sigma\rangle$ evals in one step to $\langle a', \sigma\rangle$. Alternatively, we could look at it as several cases depending on which of a_1 and a_2 are numerals. For the particular way in which we have defined $\rightarrow_1$, the form of a_1 and a_2 uniquely determine which rule was applicable to $\langle a, \sigma\rangle$. Thus, the two views are essentially equivalent. The cases are:

$a_1 \notin$ **Num:** In this case $\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle$ because the following rule applied (by plugging in a_1 for a_0 and a_2 for a_1):

$$\frac{\langle a_0, \sigma\rangle \rightarrow_1 \langle a_0', \sigma\rangle}{\langle a_0 + a_1, \sigma\rangle \rightarrow_1 \langle a_0' + a_1, \sigma\rangle}$$

So, a' must be of the form $a_1' + a_2$, for some a_1' such that $\langle a_1, \sigma\rangle \rightarrow_1 \langle a_1', \sigma\rangle$. A supposition of the Lemma was that: $\langle a', \sigma\rangle \rightarrow n$. So, by the definition of $\rightarrow$, it must be the case that there exist $n_1, n_1 \in$ **Num**, such that $\langle a_1', \sigma\rangle \rightarrow n_1$, $\langle a_2, \sigma\rangle \rightarrow n_2$, and $n_1 + n_2 = n$. So by induction, (since a_1 is a subterm of a, $\langle a_1, \sigma\rangle \rightarrow_1 \langle a_1', \sigma\rangle$ and $\langle a_1', \sigma\rangle \rightarrow n_1$), we have that $\langle a_1, \sigma\rangle \rightarrow n_1$. Finally, by definition of $\rightarrow$, and since $\langle a_1, \sigma\rangle \rightarrow n_1$, $\langle a_2, \sigma\rangle \rightarrow n_2$, and $n_1 + n_2 = n$, we have that $\langle a, \sigma\rangle \rightarrow n$, exactly as required.

This case was done in much more explicit detail than is normally required in the presentation of a proof. For any presentation to be adequate, however, enough information must be given to make it very simple to generate this level of careful detail. The last two cases should give you a better idea of the level of detail we expect from your proof presentation.

$a_1 \equiv n_1 \in$ **Num**, $a_2 \notin$ **Num:** Noting that $\langle n_1, \sigma\rangle \not\rightarrow_1$, so, a' must be of the form $n_1 + a_2'$ (we can abbreviate "a' must be of the form $n_1 + a_2'$" by "$a' \equiv n_1 + a_2'$"), where $\langle a_2, \sigma\rangle \rightarrow_1 \langle a_2', \sigma\rangle$. We finish up similarly to the preceding case. Since $\langle a', \sigma\rangle \rightarrow n$, we have that $\langle n_1, \sigma\rangle \rightarrow n_1$ and $\langle a_2', \sigma\rangle \rightarrow n_2$, where $n_1 + n_2 = n$. By induction, $\langle a_2, \sigma\rangle \rightarrow n_2$. So, finally, $\langle a, \sigma\rangle \rightarrow n$.

$a_1 \equiv n_1 \in$ **Num**, $a_2 \equiv n_2 \in$ **Num:** This case is trivial, as $\langle a, \sigma\rangle \rightarrow_1 \langle n, \sigma\rangle$. We must also have that $n_1 + n_2 = n$ (by defn of $\rightarrow$), and finally we also must have $\langle a, \sigma\rangle \rightarrow n$ (again, by definition of $\rightarrow$).

■

We now have proven the key lemma. But let us remember our goal. We wish to show that:

$$\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle \text{ implies } \langle a, \sigma\rangle \rightarrow n.$$

We prove this by induction on the definition of $\rightarrow_1^*$ as the **reflexive transitive closure** of $\rightarrow_1$.

Base Case: Suppose $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$ because $\langle a, \sigma\rangle \equiv \langle n, \sigma\rangle$. Then $\langle a, \sigma\rangle \rightarrow n$ by definition of $\rightarrow$.

Induction Step: Suppose that $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$ because $\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle$ and $\langle a', \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$. By induction: $\langle a', \sigma\rangle \rightarrow n$. So, since $\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle$, we can use the Lemma to obtain: $\langle a, \sigma\rangle \rightarrow n$.

2 Proving the "if" direction

We now prove the "if" direction ($\Leftarrow$). So, we must show that for an arbitrary $a \in$ **Aexp**, an arbitrary $n \in$ **Num**, and an arbitrary $\sigma \in \Sigma$, if it is the case that $\langle a, \sigma\rangle \rightarrow n$, then it is also the case that $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$.

This direction is much simpler than the other, and is proven by another induction on the structure of a.

Base Cases: $a \equiv n$: Trivial.

$a \equiv X$: Since $\langle a, \sigma\rangle \rightarrow n$, by the definition of $\rightarrow$ we have $n = \sigma(X)$. By the definition of $\rightarrow_1$ we then get $\langle a, \sigma\rangle \rightarrow_1 \langle n, \sigma\rangle$. Finally by the definition of $\rightarrow_1^*$ we get $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$.

Inductive Case: The inductive cases are for $a \equiv a_1 + a_2$, $a \equiv a_1 - a_2$, and $a \equiv a_1 - a_2$. We do the case of $a \equiv a_1 + a_2$, the others follow similarly.

Since $\langle a, \sigma\rangle \rightarrow n$, we have $\langle a_1, \sigma\rangle \rightarrow n_1$, and $\langle a_2, \sigma\rangle \rightarrow n_2$ by the definition of $\rightarrow$. By induction, we have $\langle a_1, \sigma\rangle \rightarrow_1^* \langle n_1, \sigma\rangle$, and $\langle a_2, \sigma\rangle \rightarrow_1^* \langle n_2, \sigma\rangle$, where $n_1 + n_2 = n$. We now make the following Remark which gets us most of the way through this case.

Remark 1. If $\langle a_1, \sigma\rangle \rightarrow_1 \langle a_1', \sigma\rangle$ then $\langle a_1 + a_2, \sigma\rangle \rightarrow_1 \langle a_1' + a_2, \sigma\rangle$ (by the definition of $\rightarrow_1$).

Using the Remark, is is a simple induction on the definition of $\rightarrow_1^*$ to show that $\langle a_1, \sigma\rangle \rightarrow_1^* \langle a_1', \sigma\rangle$ implies $\langle a_1 + a_2, \sigma\rangle \rightarrow_1^* \langle a_1' + a_2, \sigma\rangle$.

So, we have $\langle a_1 + a_2, \sigma\rangle \rightarrow_1^* \langle n_1 + a_2, \sigma\rangle$. A similar argument lets us conclude $\langle n_1 + a_2, \sigma\rangle \rightarrow_1^* \langle n_1 + n_2, \sigma\rangle$ from $\langle a_2, \sigma\rangle \rightarrow_1^* \langle n_2, \sigma\rangle$. Finally, by the definition of $\rightarrow_1$ (and the fact that $n = n_1 + n_2$), we have that $\langle n_1 + n_2, \sigma\rangle \rightarrow_1 \langle n, \sigma\rangle$. Putting all this together, we have:

$$\langle a, \sigma\rangle \equiv \langle a_1 + a_2, \sigma\rangle \rightarrow_1^* \langle n_1 + a_2, \sigma\rangle \rightarrow_1^* \langle n_1 + n_2, \sigma\rangle \rightarrow_1 \langle n, \sigma\rangle,$$

and so $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$, exactly as required.

Problem Set 3

Due: 4 October 1991.

Reading for the Problem Set. Winskel through §4.3.

Reading for Lectures. Winskel through §5.3.

Problem 1. Let FooBar be the following **IMP** command:

while $(X \geq 1) \wedge (Y \geq 1)$ **do**
  if $Y = 1$
    then $X := X - 1;$
      $Y := 2 \times X;$
      $Z := Z + 1$
    else $Y := Y - 1;$
      $Z := Z + 1$

Prove that FooBar terminates if X, Y, and Z are initially positive. In other words, show that for all states σ:

$$\sigma(X) \geq 1 \;\&\; \sigma(Y) \geq 1 \;\&\; \sigma(Z) \geq 1 \Rightarrow \exists \sigma'.\langle \text{FooBar}, \sigma \rangle \rightarrow \sigma'.$$

Problem 2. Consider the following inductive definition of a subset M of the *natural numbers* $\mathbf{N} = \{0, 1, 2, 3, \ldots\}$:

(i) $2 \in M$,
(ii) if $n \in M$, then $n^2 \in M$,
(iii) if $n, m \in M$, then $(n \cdot m) \in M$.

2(a). Prove by induction on the definition of M that

$$M = \{2^k \mid k \geq 1\} .$$

Let $f : M \rightarrow \mathbf{N}$ be any (possibly partial) function. Then f is said to be a *counter function* if

(i) $f(2) = 1$,
(ii) $f(n^2) = 2 \cdot f(n)$,
(iii) $f(n \cdot m) = f(n) + f(m)$.

2(b). Define $f_1 : M \to \mathbf{N}$ to be

$$f_1(2^k) = k.$$

Prove that f_1 is a counter function. (Hint: Induction on the definition of M.)

2(c). Prove that f_1 is the ***unique*** counter function, *i.e.*, if f_2 is any counter function, then $f_1(x) = f_2(x)$ for all $x \in M$.

2(d). Consider the function $g : M \to \mathbf{N}$ defined inductively by:

(i) $g(2) = 1$,

(ii) $g(n^2) = 5$,

(iii) $g(n \cdot m) = 10$.

Carefully prove that $1 = 0$. Explain why this contradiction occurs here, but not for counter functions.

Problem 3. (The exercise on page 38 of Winskel) Give definitions of $loc(a)$, $loc(b)$, and $loc_R(c)$, the sets of locations which appear in arithmetic expressions a, boolean expressions b, and the right-hand sides of assignments in commands c.

Problem Set 1 Solutions

1 General Information

This handout includes some of the best solutions submitted by students for Problem Set 1. These solutions are a good representation of the level of detail expected.

Problem 0. The solution to this problem appears on the next page.

Problem 1. The desired solution for this problem appeared in Handout 9. For convenience, we repeat the rules here as well. There are several things to notice about the definition. First, $\rightarrow_1$ for commands should be defined in terms of $\rightarrow_1$ for **Aexps** and for **Bexps**. It should not be defined in terms of $\rightarrow$. Why not? If a command has an Aexp within it that takes 5 steps to evaluate to a numeral, it should take that command at least 5 steps to execute. Even more fundamentally, $\rightarrow_1$ on commands should not be defined in terms of $\rightarrow$ on commands.

Two further comments: The $\rightarrow_1$ rules for **Aexps** were all of the form $\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle$. Note the pair on the right hand side. There were no rules allowing us to conclude statements of the form: $\langle a, \sigma\rangle \rightarrow_1 n$. Thus any rule with a premise of that form will never be able to apply. The same is the case for **Bexps**. Thus the discussion at the bottom of page 25 in Winskel is a little misleading when it discusses the choice as what is regarded as a single step. The idea here was that in 3 limited situations, it could be possible to short-circuit one step. They are the following situations:

$$\frac{\langle a, \sigma\rangle \rightarrow_1 \langle n, \sigma\rangle}{\langle X := a, \sigma\rangle \rightarrow_1 \sigma[n/X]}$$

$$\frac{\langle b, \sigma\rangle \rightarrow_1 \langle \mathbf{true}, \sigma\rangle}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \rightarrow_1 \langle c_0, \sigma\rangle}$$

$$\frac{\langle b, \sigma\rangle \rightarrow_1 \langle \mathbf{false}, \sigma\rangle}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \rightarrow_1 \langle c_1, \sigma\rangle}$$

Notice that even with these extra rules, you still need all of the others. Why? When giving our official definition of the rules for $\rightarrow_1$ we choose not to include the above 3 rules because adding them breaks one nice property of our definition of $\rightarrow_1$. Specifically Problem 1 on Problem Set 2 would not be true.

Alejandro Medina
6.044J/18.423J

PS# 1
Exercise ∅

9/18/91

(2)

$\sigma' \equiv \sigma[4/N][2/M]$, $\sigma'' \equiv \sigma[2/N][2/M]$

$$\dfrac{\dfrac{\langle M,\sigma''\rangle \to 2 \quad \langle N,\sigma''\rangle \to 2}{\langle (M=N),\sigma''\rangle \to true}}{\langle \neg(M=N),\sigma''\rangle \to false}$$

$$\dfrac{\dfrac{\dfrac{\langle M,\sigma'\rangle \to 2 \quad \langle N,\sigma'\rangle \to 4}{\langle (M=N),\sigma'\rangle \to false}}{\langle \neg(M=N),\sigma'\rangle \to true} \quad \dfrac{\dfrac{\langle M,\sigma'\rangle \to 2 \quad \langle N,\sigma'\rangle \to 4}{\langle M\le N,\sigma'\rangle \to true} \quad \dfrac{\dfrac{\langle N,\sigma'\rangle \to 4 \quad \langle M,\sigma'\rangle \to 2}{\langle N-M,\sigma'\rangle \to 2}}{\langle N:=N-M,\sigma'\rangle \to \sigma''}}{\langle c,\sigma'\rangle \to \sigma''} \quad \dfrac{(*)}{\langle Euclid,\sigma''\rangle \to \sigma''}}{\langle Euclid,\sigma'\rangle \to \sigma''}$$

$$\dfrac{\dfrac{\dfrac{\langle M,\sigma[4/N]\rangle \to 6 \quad \langle N,\sigma[4/N]\rangle \to 4}{\langle (M=N),\sigma[4/N]\rangle \to false}}{\langle \neg(M=N),\sigma[4/N]\rangle \to true} \quad \dfrac{\dfrac{\langle M,\sigma[4/N]\rangle \to 6 \quad \langle N,\sigma[4/N]\rangle \to 4}{\langle M\le N,\sigma[4/N]\rangle \to false} \quad \dfrac{\dfrac{\langle M,\sigma[4/N]\rangle \to 6 \quad \langle N,\sigma[4/N]\rangle \to 4}{\langle M-N,\sigma[4/N]\rangle \to 2}}{\langle M:=M-N,\sigma[4/N]\rangle \to \sigma[4/N][2/M]}}{\langle c,\sigma[4/N]\rangle \to \sigma[4/N][2/M]} \quad \dfrac{(*)}{\langle Euclid,\sigma[4/N][2/M]\rangle \to \sigma''}}{\langle Euclid,\sigma[4/N]\rangle \to \sigma''}$$

(Underlying case) ....

$$\langle Euclid,\sigma\rangle \longrightarrow \sigma[2/M][2/N] \equiv \sigma''$$

The following is the full set of $\rightarrow_1$ rules for commands which we were looking for.

Atomic Commands:

$$\langle \mathbf{skip}, \sigma \rangle \rightarrow_1 \sigma$$

$$\frac{\langle a, \sigma \rangle \rightarrow_1 \langle a', \sigma \rangle}{\langle X := a, \sigma \rangle \rightarrow_1 \langle X := a', \sigma \rangle}$$

$$\langle X := n, \sigma \rangle \rightarrow_1 \sigma[n/X]$$

Sequencing:

$$\frac{\langle c_0, \sigma \rangle \rightarrow_1 \langle c_0', \sigma' \rangle}{\langle c_0; c_1, \sigma \rangle \rightarrow_1 \langle c_0'; c_1, \sigma' \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \rightarrow_1 \sigma'}{\langle c_0; c_1, \sigma \rangle \rightarrow_1 \langle c_1, \sigma' \rangle}$$

Conditionals:

$$\frac{\langle b, \sigma \rangle \rightarrow_1 \langle b', \sigma \rangle}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma \rangle \rightarrow_1 \langle \mathbf{if}\, b'\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma \rangle}$$

$$\langle \mathbf{if}\, \mathbf{true}\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma \rangle \rightarrow_1 \langle c_0, \sigma \rangle$$

$$\langle \mathbf{if}\, \mathbf{false}\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma \rangle \rightarrow_1 \langle c_1, \sigma \rangle$$

While-loops:

$$\langle \mathbf{while}\, b\, \mathbf{do}\, c, \sigma \rangle \rightarrow_1 \langle \mathbf{if}\, b\, \mathbf{then}(c; \mathbf{while}\, b\, \mathbf{do}\, c)\, \mathbf{else}\, \mathbf{skip}, \sigma \rangle$$

Problem 2. The intended goal of this problem was to make side-effects uniformly available within all **Aexps**. Essentially we wanted you to introduce a construct like SEQUENCE in SCHEME, c **resultis** a should behave essentially like (SEQUENCE c a) would in scheme. c **resultis** a can be used anywhere that you can use an **Aexp**, including within another **resultis** construct.

A solution to this problem appears on the next page.

Michael G. Sheldon
6.044J/18.423J
PS1
Problem 2
20 SEP 91

2. IMP-SIDE-EFFECTS:

Aexp: $a ::= n \mid X \mid c\ \underline{\text{resultis}}\ a \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$

Bexp: $b ::= \text{true} \mid \text{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$

Com: $c ::= \underline{\text{skip}} \mid X := a \mid c_0 ; c_1 \mid \underline{\text{if}}\ b\ \underline{\text{then}}\ c_0\ \underline{\text{else}}\ c_1 \mid \underline{\text{while}}\ b\ \underline{\text{do}}\ c$

Aexp:

$$\langle X, \sigma \rangle \to \langle \sigma(X), \sigma \rangle$$

$$\langle n, \sigma \rangle \to \langle n, \sigma \rangle$$

$$\frac{\langle a_0, \sigma \rangle \to \langle n_0, \sigma'' \rangle \quad \langle a_1, \sigma'' \rangle \to \langle n_1, \sigma' \rangle}{\langle a_0 + a_1, \sigma \rangle \to \langle n, \sigma' \rangle} \quad \text{where } n = n_0 \text{ plus } n_1$$

$$\frac{\langle a_0, \sigma \rangle \to \langle n_0, \sigma'' \rangle \quad \langle a_1, \sigma'' \rangle \to \langle n_1, \sigma' \rangle}{\langle a_0 - a_1, \sigma \rangle \to \langle n, \sigma' \rangle} \quad \text{where } n = n_0 \text{ sub } n_1$$

$$\frac{\langle a_0, \sigma \rangle \to \langle n_0, \sigma'' \rangle \quad \langle a_1, \sigma'' \rangle \to \langle n_1, \sigma' \rangle}{\langle a_0 \times a_1, \sigma \rangle \to \langle n, \sigma' \rangle} \quad \text{where } n = n_0 \text{ mult } n_1$$

$$\frac{\langle c, \sigma \rangle \to \sigma'' \quad \langle a_0, \sigma'' \rangle \to \langle n, \sigma' \rangle}{\langle c\ \underline{\text{resultis}}\ a_0, \sigma \rangle \to \langle n, \sigma' \rangle}$$

Bexp:

$$\frac{\langle a_0, \sigma \rangle \to \langle n, \sigma'' \rangle \quad \langle a_1, \sigma'' \rangle \to \langle m, \sigma' \rangle}{\langle a_0 = a_1, \sigma \rangle \to \langle t, \sigma' \rangle} \quad \text{where } t \equiv n \text{ equals } m$$

$$\frac{\langle a_0, \sigma \rangle \to \langle n, \sigma'' \rangle \quad \langle a_1, \sigma'' \rangle \to \langle m, \sigma' \rangle}{\langle a_0 \leq a_1, \sigma \rangle \to \langle t, \sigma' \rangle} \quad \text{where } t \equiv n \text{ at most } m$$

$$\frac{\langle b, \sigma \rangle \to \langle \text{true}, \sigma' \rangle}{\langle \neg b, \sigma \rangle \to \langle \text{false}, \sigma' \rangle} \qquad \frac{\langle b, \sigma \rangle \to \langle \text{false}, \sigma' \rangle}{\langle \neg b, \sigma \rangle \to \langle \text{true}, \sigma' \rangle}$$

Bexp:

$$\frac{\langle b_0, \sigma\rangle \to \langle t_0, \sigma''\rangle \quad \langle b_1, \sigma''\rangle \to \langle t_1, \sigma'\rangle}{\langle b_0 \wedge b_1, \sigma\rangle \to \langle t, \sigma'\rangle} \quad \text{where } t \equiv t_0 \text{ and } t_1$$

$$\frac{\langle b_0, \sigma\rangle \to \langle t_0, \sigma''\rangle \quad \langle b_1, \sigma''\rangle \to \langle t_1, \sigma'\rangle}{\langle b_0 \vee b_1, \sigma\rangle \to \langle t, \sigma'\rangle} \quad \text{where } t \equiv t_0 \text{ or } t_1$$

Com:

$$\langle \underline{\text{skip}}, \sigma\rangle \to \sigma$$

$$\frac{\langle a, \sigma\rangle \to \langle m, \sigma'\rangle}{\langle X := a, \sigma\rangle \to \sigma'[m/X]}$$

$$\frac{\langle c_0, \sigma\rangle \to \sigma'' \quad \langle c_1, \sigma''\rangle \to \sigma'}{\langle c_0; c_1, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to \langle \text{true}, \sigma''\rangle \quad \langle c_0, \sigma''\rangle \to \sigma'}{\langle \underline{\text{if}}\ b\ \underline{\text{then}}\ c_0\ \underline{\text{else}}\ c_1, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to \langle \text{false}, \sigma''\rangle \quad \langle c_1, \sigma''\rangle \to \sigma'}{\langle \underline{\text{if}}\ b\ \underline{\text{then}}\ c_0\ \underline{\text{else}}\ c_1, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to \langle \text{false}, \sigma'\rangle}{\langle \underline{\text{while}}\ b\ \underline{\text{do}}\ c, \sigma\rangle \to \sigma'}$$

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

Problem Set 2 Solutions

Problem 1. [Deterministic Rewriting] Let γ denote a command configuration of the while programming language **IMP**, and δ denote either a command configuration or a state.

1(a). Prove by structural induction that for every γ, there is exactly one δ such that $\gamma \rightarrow_1 \delta$. Briefly comment on where structural induction would break down for a similar proof about the "evaluates to" relation.

Solution: It was announced that we may assume that for all $a \in (\mathbf{Aexp} - \mathbf{Num})$ there is exactly one a' such that $\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle$. And similarly, for all $b \in (\mathbf{Bexp} - \{\mathbf{true}, \mathbf{false}\})$ there is exactly one b' such that $\langle b, \sigma\rangle \rightarrow_1 \langle b', \sigma\rangle$.

Let γ be $\langle c, \sigma\rangle$. We now have several cases depending on the structure of c:

[$c \equiv$ **skip** .] If $\langle c, \sigma\rangle \rightarrow_1 \delta$, then it must have been due to the axiom (*i.e.*, rule with no antecedents):

$$\langle \mathbf{skip}, \sigma\rangle \rightarrow_1 \sigma$$

This axiom is always applicable, and in exactly one way. So δ exists, and it must be σ.

[$c \equiv\ X := a$.] We again have subcases based on a.

[$a \notin$ **Num** .] By the announcement, we know that there is exactly one a' such that $\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle$. Moreover there is exactly one form of rule which can apply in this case, namely:

$$\frac{\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle}{\langle X := a, \sigma\rangle \rightarrow_1 \langle X := a', \sigma\rangle}$$

Thus the only configuration to which $\langle X := a, \sigma\rangle$ can rewrite in one step is $\langle X := a', \sigma\rangle$. Moreover, this rewriting can always occur.

[$a \equiv n \in$ **Num** .] We have already established (in class, and in another handout) that $\langle n, \sigma\rangle$ cannot rewrite to anywhere. Thus the only rule which can apply is the axiom:

$$\langle X := n, \sigma\rangle \rightarrow_1 \sigma[n/X]$$

Moreover this axiom can always apply to c, giving $\delta = \sigma[n/X]$.

[$c \equiv c_0; c_1$.] Here it looks like we might have a problem. It looks like two rules might apply. But by induction (applied to c_0) there is exactly one δ_0 such that $\langle c_0, \sigma\rangle \rightarrow_1 \delta_0$. If δ_0 is of the form $\langle c_0', \sigma'\rangle$, then exactly one rule can apply in this case:

$$\frac{\langle c_0, \sigma\rangle \rightarrow_1 \langle c_0', \sigma'\rangle}{\langle c_0; c_1, \sigma\rangle \rightarrow_1 \langle c_0'; c_1, \sigma'\rangle}$$

Moreover, this rule can always apply, giving $\delta \equiv \langle c_0'; c_1, \sigma'\rangle$.

On the other hand, if δ_0 is of the form σ', then there is still exactly one rule which can apply, this time it is:

$$\frac{\langle c_0, \sigma\rangle \rightarrow_1 \sigma'}{\langle c_0; c_1, \sigma\rangle \rightarrow_1 \langle c_1, \sigma'\rangle}$$

Moreover this rule can always apply, giving $\delta \equiv \langle c_1, \sigma'\rangle$.

[$c \equiv \textbf{if}\, b\, \textbf{then}\, c_0\, \textbf{else}\, c_1$.] We have three subcases depending on the form of b.

[$b \notin \{\textbf{true}, \textbf{false}\}$.] This case works similar to that of $c \equiv X := a$, with $a \notin \textbf{Num}$.

[$b \equiv \textbf{true}$.] In this case exactly 1 rule can apply (since $\langle \textbf{true}, \sigma\rangle \not\rightarrow_1$). Thus $\delta \equiv \langle c_0, \sigma\rangle$.

[$b \equiv \textbf{false}$.] Similar to case of $b \equiv \textbf{true}$.

[$c \equiv \textbf{while}\, b\, \textbf{do}\, c'$.] This case is trivial, but I'll do it anyway. In this case, only one rule can apply. It must be that

$$\delta \equiv \langle \textbf{if}\, b\, \textbf{then}(c; \textbf{while}\, b\, \textbf{do}\, c)\, \textbf{else}\, \textbf{skip}, \sigma\rangle.$$

What breaks down in doing structural induction for a similar proof about the "evaluates to" relation, is the case of **while**. Notice that for this proof there was only one configuration to which a **while** loop could rewrite, so there was no need to use induction there. For "evaluates to" however, the behavior of a **while**-loop in some state can depend on the behavior of that same **while**-loop in some other state. Structural induction only works when the cases for big terms only need to use the induction hypothesis for their subterms. For "evaluates to" we would need to know that the conclusion held for $\textbf{while}\, b\, \textbf{do}\, c$, in order to prove that it held for $\textbf{while}\, b\, \textbf{do}\, c$, which is clearly circular.

1(b). Conclude that there is a partial function

$$\textit{eval} : [\text{command configurations}] \rightarrow \Sigma$$

such that for all states $\sigma \in \Sigma$

$$\gamma \rightarrow_1^* \sigma \text{ iff } \textit{eval}(\gamma) = \sigma.$$

Solution: This is essentially asking you to prove that for all command configurations γ there is at most one state σ, such that $\gamma \rightarrow_1^* \sigma$. Then $eval(\gamma)$ can be defined to be this unique σ if it exists, otherwise $eval(\gamma)$ is undefined.

So we must prove that if $\gamma \rightarrow_1^* \sigma$ and $\gamma \rightarrow_1^* \sigma'$, then $\sigma = \sigma'$. The proof is by induction on the definition of $\gamma \rightarrow_1^* \sigma$ (as the reflexive transitive closure of $\rightarrow_1$).

The base case is that $\gamma \rightarrow_1^* \sigma$ because $\gamma \equiv \sigma$. But this is impossible, since γ is a command configuration and σ is a state, so there is nothing to prove.

The inductive step is the case that $\gamma \rightarrow_1^* \sigma$ because $\gamma \rightarrow_1 \delta$ and $\delta \rightarrow_1^* \sigma$ for some δ.

Suppose that also $\gamma \rightarrow_1^* \sigma'$. Since γ is not a state, we must have $\gamma \rightarrow_1 \delta'$ and $\delta' \rightarrow_1^* \sigma'$ for some δ'. By part (a), $\delta = \delta'$.

If δ is a state, then because there is no $\rightarrow_1$ rule for rewriting states, it must be that $\sigma = \delta$ and $\sigma' = \delta'$, so $\sigma = \sigma'$. If δ is a command configuration, then we have $\delta \rightarrow_1^* \sigma$ and also $\delta \rightarrow_1^* \sigma$, so $\sigma' = \sigma$ by induction hypothesis. ■

Problem 2. This problem is based on the exercise in Winskel, p. 47, proving equivalence of rewriting and evaluation semantics of **IMP** commands. We have broken the problem up into several "independent subproblems." If you are unable to do a part, go on to the next part. For all later parts, you may assume that you have done all of the other other parts correctly.

Handout 8 given out on September 20, and handout 9 which will be given out on September 23, will form our official definition of the relation $\rightarrow_1$ (the one step execution relation). The goal of this problem is to prove the following Theorem:

$$\forall \sigma, \sigma'.[\langle c, \sigma\rangle \rightarrow_1^* \sigma' \text{ iff } \langle c, \sigma\rangle \rightarrow \sigma'].$$

For this problem, you may assume that for $a \in \mathbf{Aexp}$, $n \in Num$, $b \in \mathbf{Bexp}, t \in \{\mathtt{true}, \mathtt{false}\}$:

$$\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle \text{ iff } \langle a, \sigma\rangle \rightarrow n$$

and

$$\langle b, \sigma\rangle \rightarrow_1^* \langle t, \sigma\rangle \text{ iff } \langle b, \sigma\rangle \rightarrow t$$

2(a). Prove

$$\forall \sigma, \sigma'.[\langle c_0; c_1, \sigma\rangle \rightarrow_1^* \sigma' \Rightarrow \exists \sigma''.\langle c_0, \sigma\rangle \rightarrow_1^* \sigma'' \,\&\, \langle c_1, \sigma''\rangle \rightarrow_1^* \sigma']$$

by induction on the definition of $\rightarrow_1^*$ as the *transitive closure* of $\rightarrow_1$, in the execution of $\langle c_0; c_1, \sigma\rangle \rightarrow_1^* \sigma'$.

Proof: The base case is that of $\langle c_0; c_1, \sigma\rangle \rightarrow_1^* \sigma'$ because $\langle c_0; c_1, \sigma\rangle \equiv \sigma'$. But this is impossible, so there is nothing to prove.

For the inductive step we must consider all γ's such that $\langle c_0; c_1, \sigma\rangle \rightarrow_1 \gamma$, and $\gamma \rightarrow_1^* \sigma'$. Looking at the rules which could have applied to obtain $\langle c_0; c_1, \sigma\rangle \rightarrow_1 \gamma$, we have 2 cases.

- γ is of the form $\langle c_0'; c_1, \sigma_0\rangle$, and $\langle c_0, \sigma\rangle \rightarrow_1 \langle c_0', \sigma_0\rangle$. Then by the induction hypothesis (applied to $\langle c_0'; c_1, \sigma\rangle$), we have that there is a σ'' such that $\langle c_0', \sigma_0\rangle \rightarrow_1^* \sigma''$ and $\langle c_1, \sigma''\rangle \rightarrow_1^* \sigma'$. Since $\langle c_0, \sigma\rangle \rightarrow_1 \langle c_0', \sigma_0\rangle$, we have that $\langle c_0, \sigma\rangle \rightarrow_1^* \sigma''$, as required.
- γ is of the form $\langle c_1, \sigma_0\rangle$, and $\langle c_0, \sigma\rangle \rightarrow_1 \sigma_0$. In this case we're done. Simply let σ'' be σ_0.

■

2(b). Prove

$$\forall \sigma, \sigma', \sigma''.[\langle c_0, \sigma\rangle \rightarrow_1^* \sigma'' \ \&\ \langle c_1, \sigma''\rangle \rightarrow_1^* \sigma' \Rightarrow \langle c_0; c_1, \sigma\rangle \rightarrow_1^* \sigma']$$

again by induction on the definition of $\rightarrow_1^*$ as the transitive closure of $\rightarrow_1$, this time using the execution of $\langle c_0, \sigma\rangle \rightarrow_1^* \sigma''$.

Proof: Again the basis step holds trivially.

For the induction step. Suppose $\langle c_0, \sigma\rangle \rightarrow_1 \gamma$ and $\gamma \rightarrow_1^* \sigma''$. Moreover, suppose $\langle c_1, \sigma''\rangle \rightarrow_1 \sigma'$. We must show that $\langle c_0; c_1, \sigma\rangle \rightarrow_1^* \sigma'$. We again have 2 cases.

- γ is of the form $\langle c_0', \sigma_0\rangle$. Then by the induction hypothesis (applied to $\langle c_0', \sigma_0\rangle$), $\langle c_0'; c_1, \sigma_0\rangle \rightarrow_1^* \sigma'$. But since $\langle c_0, \sigma\rangle \rightarrow_1 \langle c_0', \sigma_0\rangle$ we have by the definition of $\rightarrow_1$ that $\langle c_0; c_1, \sigma\rangle \rightarrow_1 \langle c_0'; c_1, \sigma_0\rangle$. Combining things gives us $\langle c_0; c_1, \sigma\rangle \rightarrow_1^* \sigma'$.
- γ is of the form σ''. The result then holds trivially, since $\langle c_0, \sigma\rangle \rightarrow_1 \sigma''$ implies that $\langle c_0; c_1, \sigma\rangle \rightarrow_1 \langle c_1, \sigma''\rangle$. Combining this with our original presumption of $\langle c_1, \sigma''\rangle \rightarrow_1^* \sigma'$ gives us our desired result.

■

From parts (a) and (b) we can then conclude the Lemma:

$$\langle c_0; c_1, \sigma\rangle \rightarrow_1^* \sigma' \text{ iff } \exists \sigma''.\langle c_0, \sigma\rangle \rightarrow_1^* \sigma'' \ \&\ \langle c_1, \sigma''\rangle \rightarrow_1^* \sigma',$$

for all commands c_0, c_1, and states σ, σ'. This Lemma will be essential to finish proving the Theorem.

2(c). Prove

$$\forall \sigma, \sigma'.[\langle c, \sigma\rangle \rightarrow_1^* \sigma' \Rightarrow \langle c, \sigma\rangle \rightarrow \sigma']$$

by structural induction on c. Do all cases except for that of c being a **while** loop.

Proof: Suppose $\langle c, \sigma\rangle \rightarrow_1^* \sigma'$. We must then show that $\langle c, \sigma\rangle \rightarrow \sigma'$. We do this by induction on the structure of c. We will break the proof down by cases on the structure of c.

[$c \equiv$ **skip**.] Obvious.

[$c \equiv X := a$.] It is a simple induction on the definition of $\rightarrow_1^*$ to show that if $\langle X := a, \rightarrow\rangle_1^* \sigma'$ then there exists an n such that $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$ and $\sigma' = \sigma[n/X]$. One of our premisses in the problem statement, was that if $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$ then $\langle a, \sigma\rangle \rightarrow n$. Since we have $\langle a, \sigma\rangle \rightarrow_1^* \langle n, \sigma\rangle$ we can conclude that $\langle a, \sigma\rangle \rightarrow n$. Finally, $\langle X := a, \sigma\rangle \rightarrow \sigma[n/X]$ follows simply by the definition of $\rightarrow$.

[$c \equiv c_0; c_1$.] Since $\langle c_0; c_1, \sigma\rangle \rightarrow_1^* \sigma'$, by part (a) there exists a σ'' such that $\langle c_0, \sigma\rangle \rightarrow_1^* \sigma''$ and $\langle c_1, \sigma''\rangle \rightarrow_1^* \sigma'$. By the induction hypothesis $\langle c_0, \sigma\rangle \rightarrow \sigma''$, and $\langle c_1, \sigma''\rangle \rightarrow \sigma$. Finally, by the definition of $\rightarrow$ we have $\langle c_0; c_1, \sigma\rangle \rightarrow \sigma'$.

[$c \equiv$ **if** b **then** c_0 **else** c_1.] The key step is to prove by induction on the definition of $\rightarrow_1^*$, that:

> If $\langle$**if** b **then** c_0 **else** $c_1, \sigma\rangle \rightarrow_1^* \sigma$, then either $\langle b, \sigma\rangle \rightarrow_1^*$ **true** and $\langle c_0, \sigma\rangle \rightarrow_1^* \sigma'$, or $\langle b, \sigma\rangle \rightarrow_1^*$ **false** and $\langle c_1, \sigma\rangle \rightarrow_1^* \sigma'$.

We then have two cases depending on whether $\langle b, \sigma\rangle \rightarrow_1^* \langle$**true**$, \sigma\rangle$ or $\langle b, \sigma\rangle \rightarrow_1^* \langle$**false**$, \sigma\rangle$. In the **true** case we use one of our premisses in the problem statement to get $\langle b, \sigma\rangle \rightarrow$ **true**, and we use the induction hypothesis to get $\langle c_0, \sigma\rangle \rightarrow \sigma'$ from $\langle c_0, \sigma\rangle \rightarrow_1^* \sigma'$. We get the end result of $\langle$**if** b **then** c_0 **else** $c_1, \sigma\rangle \rightarrow \sigma'$ from the definition of $\rightarrow$. The **false** case goes through similarly.

[$c \equiv$ **while** b **do** c'.] See part (d).

■

2(d). Do the case of c a **while** loop for the proof of part (c). This can be proven by induction on the length of the computation.

Proof: Let $w \equiv \textbf{while}\, b\, \textbf{do}\, c'$.

Since this is actually a case from the preceeding proof, we are allowed to invoke the induction hypothesis from there. In other words we may assume that if $\langle c', \sigma\rangle \rightarrow_1^* \sigma'$ then $\langle c', \sigma\rangle \rightarrow \sigma'$.

The base case is again trivial.

The inductive step is that:

$$\langle w, \sigma\rangle \rightarrow_1 \langle \textbf{if}\, b\, \textbf{then}(c; w)\, \textbf{else}\, \textbf{skip}, \sigma\rangle \rightarrow_1^* \sigma'$$

and,

$$\langle \textbf{if}\, b\, \textbf{then}(c; w)\, \textbf{else}\, \textbf{skip}, \sigma\rangle \rightarrow_1^* \sigma'$$

To prove that $\langle w, \sigma\rangle \rightarrow_1^* \sigma'$ we will use the sublemma used in the case for **if** in part (c). Specifically,

> If $\langle \textbf{if}\, b\, \textbf{then}\, c_0\, \textbf{else}\, c_1, \sigma\rangle \rightarrow_1^* \sigma'$ then either $\langle b, \sigma\rangle \rightarrow_1^* \langle \textbf{true}, \sigma\rangle$ and $\langle c_0, \sigma\rangle \rightarrow_1^* \sigma'$ or $\langle b, \sigma\rangle \rightarrow_1^* \langle \textbf{false}, \sigma\rangle$ and $\langle c_1, \sigma\rangle \rightarrow_1^* \sigma'$.

This gives us two cases.

- Here we have $\langle b, \sigma\rangle \rightarrow_1^* \langle \textbf{true}, \sigma\rangle$ and $\langle c'; w, \sigma\rangle \rightarrow_1^* \sigma'$. By the premis of the problem we have that $\langle b, \sigma\rangle \rightarrow \textbf{true}$. By part (a), we have that there exists a σ'' such that $\langle c', \sigma\rangle \rightarrow_1^* \sigma''$ and $\langle w, \sigma''\rangle \rightarrow_1^* \sigma'$. So by the induction hypothesis of part (d), we may conclude that $\langle w, \sigma''\rangle \rightarrow \sigma'$. By the induction hypothesis of part (c) (of which this is a piece of the proof), we may conclude that $\langle c, \sigma\rangle \rightarrow \sigma''$. Finally using the definition of $\rightarrow$, we get that $\langle w, \sigma\rangle \rightarrow \sigma'$.
- Here we have $\langle b, \sigma\rangle \rightarrow_1^* \langle \textbf{false}, \sigma\rangle$, and $\sigma = \sigma'$. The rest of this case follows trivially from the premiss of the problem.

■

2(e). Prove

$$\forall \sigma, \sigma'.[\langle c, \sigma\rangle \rightarrow \sigma' \Rightarrow \langle c, \sigma\rangle \rightarrow_1^* \sigma']$$

by rule induction (or by induction on derivations).

The solution to this part appears on the next page.

Parts (c), (d), and (e) give us our main goal.

2(e) So we must show for all derivations d:
if d is a derivation of $\langle c, \sigma \rangle \to \sigma'$ then $\langle c, \sigma \rangle \to_1^* \sigma'$.

We prove this by induction on the derivation d.

We break the proof down into cases based on the <u>STRUCTURE OF C!!</u>

Case $c \equiv$ skip. Trivial. (Why? def of $\to$ gives that $\sigma' = \sigma$ and we know $\langle \text{skip}, \sigma \rangle \to_1^* \sigma$.)

Case $c \equiv X := a$. Then d must be of the form:

$$\frac{\dfrac{\vdots}{\langle a, \sigma \rangle \to n}}{\langle X := a, \sigma \rangle \to \sigma[n/X]}$$

for some n. (and $\sigma' = \sigma[n/X]$).

In the problem statement we were told that we could conclude $\langle a, \sigma \rangle \to_1^* \langle n, \sigma \rangle$ from $\langle a, \sigma \rangle \to n$.
It is then a trivial induction on the definition of $\to_1^*$ to show that $\langle X := a, \sigma \rangle \to_1^* \langle X := n, \sigma \rangle$. By def of $\to_1$, $\langle X := n, \sigma \rangle \to_1 \sigma[n/X]$.
So $\langle X := a, \sigma \rangle \to_1^* \sigma[n/X]$

Case $c \equiv c_0 ; c_1$. Then d must be of the form

$$\frac{d_0 \left\{ \dfrac{\vdots}{\langle c_0, \sigma \rangle \to \sigma''} \right. \qquad d_1 \left\{ \dfrac{\vdots}{\langle c_1, \sigma'' \rangle \to \sigma'} \right.}{\langle c_0 ; c_1, \sigma \rangle \to \sigma'}$$

for some σ'.

Let d_0 + d_1 be the indicated subderivations of d.
Then by induction: (on d_0) $\langle c_0, \sigma \rangle \to_1^* \sigma''$
(on d_1) $\langle c_1, \sigma'' \rangle \to_1^* \sigma'$.

Then by part (b) of this problem, $\langle c_0 ; c_1, \sigma \rangle \to_1^* \sigma'$.

since $\langle b,\sigma\rangle \to false$, $\langle b,\sigma\rangle \to_1^* \langle false, \sigma\rangle$.

By defn of $\to_1$, $\langle w,\sigma\rangle \to_1 \langle \textit{if } b \textit{ then } c'; w \textit{ else skip}, \sigma\rangle$.
~~by the~~ I'll no longer say by a trivial induction on the definition on $\to_1^*$, but simply say,
since $\langle b,\sigma\rangle \to_1^* \langle false,\sigma\rangle$ we have: ~~or b then~~
$\langle \textit{if } b \textit{ then } c'; w \textit{ else skip}, \sigma\rangle \to_1^* \langle \textit{if false then } c'; w \textit{ else skip}, \sigma\rangle$
since $\langle \textit{if false then } c'; w \textit{ else skip}, \sigma\rangle \to_1^* \sigma$.
Putting it together gives $\langle w,\sigma\rangle \to_1^* \sigma$.

Second Subcase: d looks like

$$\frac{\dfrac{\vdots}{\langle b,\sigma\rangle \to true} \qquad d_0\left\{\dfrac{\vdots}{\langle c',\sigma\rangle \to \sigma''}\right. \qquad d_1\left\{\dfrac{\vdots}{\langle w,\sigma''\rangle \to \sigma'}\right.}{\langle w,\sigma\rangle \to \sigma'}$$

for some σ''. Let d_0, d_1 be the indicated subderivations a

As we have seen several times we can then conclude:
$\langle b,\sigma\rangle \to_1^* \langle true,\sigma\rangle$
$\langle c',\sigma\rangle \to_1^* \sigma''$ (by induction applied to d_0)
$\langle w,\sigma''\rangle \to_1^* \sigma'$ (by induction applied to d_1)
and so
$\langle c'; w,\sigma\rangle \to_1^* \sigma'$ (by part (b)).

Then by definition of $\to_1^*$
$\langle w,\sigma\rangle \to_1 \langle \textit{if } b \textit{ then } c'; w \textit{ else skip}, \sigma\rangle$
$\to_1^* \langle \textit{if true then } c'; w \textit{ else skip}, \sigma\rangle$
$\to_1 \langle c'; w, \sigma\rangle$
$\to_1^* \sigma'$,
exactly as required!

case $c \equiv$ if b then c_0 else c_1. We now have 2 subcases based on the final rule applied in d.

First subcase: d looks like

$$\frac{\begin{array}{c}\vdots\\ \hline \langle b,\sigma\rangle \to true\end{array} \qquad \left.\begin{array}{c}\vdots\\ \hline \langle c_0,\sigma\rangle \to \sigma'\end{array}\right\} d'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \to \sigma'}$$

. Let d' be the indicated subderivation of d.

In the problem statement we were told that we could conclude $\langle b,\sigma\rangle \to_1^* \langle true,\sigma\rangle$ from $\langle b,\sigma\rangle \to true$.

It is then a trivial induction on the definition to show that $\langle \text{if } b \text{ then } c_0 \text{ else } c_1,\sigma\rangle \to_1^* \langle \text{if } true \text{ then } c_0 \text{ else } c_1,\sigma\rangle$.
then by def of $\to_1$, we have $\langle \text{if } true \text{ then } c_0 \text{ else } c_1,\sigma\rangle \to_1 \langle c_0,\sigma\rangle$.
Finally by the induction hypothesis (applied to d') we have $\langle c_0,\sigma\rangle \to_1^* \sigma'$.
Putting this together gives $\langle \text{if } b \text{ then } c_0 \text{ else } c_1,\sigma\rangle \to_1^* \sigma'$.

Second subcase: d looks like

$$\frac{\begin{array}{c}\vdots\\ \hline \langle b,\sigma\rangle \to false\end{array} \qquad \begin{array}{c}\vdots\\ \hline \langle c_1,\sigma\rangle \to \sigma'\end{array}}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \to \sigma'}$$

works similar to first subcase.

case $c \equiv$ while b do c'. We now have 2 subcases based on the final rule applied in d.
Let w = while b do c'

First subcase: d looks like

$$\frac{\begin{array}{c}\vdots\\ \hline \langle b,\sigma\rangle \to false\end{array}}{\langle \text{while } b \text{ do } c',\sigma\rangle \to \sigma}$$

and so $\sigma' = \sigma$.

Problem Set 3 Solutions

Problem 1. Let FooBar be the following **IMP** command:

while $(X \geq 1) \wedge (Y \geq 1)$ **do**
  if $Y = 1$
    then $X := X - 1;$
        $Y := 2 \times X;$
        $Z := Z + 1$
    else $Y := Y - 1;$
        $Z := Z + 1$

Prove that FooBar terminates if X, Y, and Z are initially positive. In other words, show that for all states σ:

$$\sigma(X) \geq 1 \;\&\; \sigma(Y) \geq 1 \;\&\; \sigma(Z) \geq 1 \Rightarrow \exists\sigma'.\langle \text{FooBar}, \sigma\rangle \to \sigma'.$$

Solution: This problem is very similar to showing that Euclid terminates.

So we need to show that the property

$$P(\sigma) \text{ iff } \exists\sigma'.\langle \text{Foobar}, \sigma\rangle \to \sigma'$$

holds for all states σ in $T = \{\sigma \in \Sigma | \sigma(X) \geq 1 \;\&\; \sigma(Y) \geq 1 \;\&\; \sigma(Z) \geq 1\}$.

To show this we will in fact show that P holds for all states σ in the set $S = \{\sigma \in \Sigma | \sigma(X) \geq 0 \;\&\; \sigma(Y) \geq 0\}$ Notice that we have dropped the condition on Z as it was not relevant to the termination of Foobar, we have relaxed the condition on $\sigma(X)$ to $\sigma(X) \geq 0$, and we have also relaxed the condition on $\sigma(Y)$ to $\sigma(Y) \geq 0$ (you'll see why we did this near the end of the proof). Notice that $T \subseteq S$, so if P holds for all states in S it will hold for all states in T.

We show that the property holds for all states in S by well-founded induction on the relation $\prec_{\text{LEX}}$ on S, where $\prec_{\text{LEX}}$ orders states lexicographically by their values in $\sigma(X)$ and $\sigma(Y)$. In other words:

$$\sigma \prec_{\text{LEX}} \sigma' \text{ iff } \begin{array}{ll} \text{either} & \sigma(X) < \sigma'(X), \\ \text{or} & \sigma(X) = \sigma(X') \;\&\; \sigma(Y) < \sigma'(Y). \end{array}$$

It is not so obvious that $\prec_{\text{LEX}}$ is well-founded. It is, however, a simple variant on the lexicographic ordering discussed in class, and it is fairly simple to translate the argument given in class to apply to $\prec_{\text{LEX}}$. The rest of the proof is a rehashing of the proof for Euclid.

Let $\sigma \in S$. Suppose $\forall \sigma' \prec_{\text{LEX}} \sigma . P(\sigma')$. Let $x = \sigma(X)$, $y = \sigma(Y)$, and $z = \sigma(Z)$.

If $x = 0$ or $y = 0$. So, since $x = 0$ or $y = 0$, $\langle (X \geq 1) \wedge (Y \geq 1), \sigma \rangle \rightarrow \mathbf{false}$. Using its derivation we construct the derivation

$$\frac{\begin{array}{c}\vdots \\ \langle (X \geq 1) \wedge (Y \geq 1), \sigma \rangle \rightarrow \mathbf{false}\end{array}}{\langle \text{Foobar}, \sigma \rangle \rightarrow \sigma}$$

using the rule for while-loops which applies when the boolean condition evaluates to false. So in the case where $x < 1$, $\langle \text{Foobar}, \sigma \rangle \rightarrow \sigma$.

Otherwise $x \geq 1$ and $y \geq 1$. In this case $\langle (X \geq 1) \wedge (Y \geq 1), \sigma \rangle \rightarrow \mathbf{true}$. From the rules for the executions of commands, we derive $\langle c, \sigma \rangle \rightarrow \sigma''$, where c is the body of the while loop. Specifically let c be the following command:

$$\begin{array}{lll}\textbf{if} & Y = 1 & \\ & \textbf{then} & X := X - 1; \\ & & Y := 2 \times X; \\ & & Z := Z + 1 \\ & \textbf{else} & Y := Y - 1; \\ & & Z := Z + 1\end{array}$$

And where

$$\sigma'' = \begin{array}{ll}\sigma[(x-1)/X][(2 \times (x-1))/Y][(z+1)/Z] & \text{if } y = 1, \\ \sigma[(y-1)/Y][(z+1)/Z] & \text{if } y \neq 1.\end{array}$$

In either case $\sigma'' \in S$ and $\sigma \prec_{\text{LEX}} \sigma$. (Note that if we simply did induction in the set of states T, we would not be guaranteed that $\sigma'' \in T$—why? Consider the case of $\sigma(X) = 1$ and $\sigma(Y) = 1$.) Thus, by induction $P(\sigma'')$, and so $\langle \text{Foobar}, \sigma'' \rangle \rightarrow \sigma'$ for some σ'. Thus applying the other rule for while-loops we obtain

$$\frac{\dfrac{\vdots}{\langle (X \geq 1) \wedge (Y \geq 1), \sigma \rangle \rightarrow \mathbf{true}} \quad \dfrac{\vdots}{\langle c, \sigma \rangle \rightarrow \sigma''} \quad \dfrac{\vdots}{\langle \text{Foobar}, \sigma'' \rangle \rightarrow \sigma'}}{\langle \text{Foobar}, \sigma \rangle \rightarrow \sigma'}$$

a derivation of $\langle \text{Foobar}, \sigma \rangle \rightarrow \sigma'$. Therefore $P(\sigma)$.

By well-founded induction we conclude $\forall \sigma \in S . P(\sigma)$, which is sufficient to prove the desired result.

Problem 2. Consider the following inductive definition of a subset M of the *natural numbers* $\mathbf{N} = \{0, 1, 2, 3, \ldots\}$:

(i) $2 \in M$,

(ii) if $n \in M$, then $n^2 \in M$,

(iii) if $n, m \in M$, then $(n \cdot m) \in M$.

The solution to all of the parts of problem 2 appears on the next page.

2(a). Prove by induction on the definition of M that

$$M = \{2^k \mid k \geq 1\} .$$

Let $f : M \to \mathbf{N}$ be any (possibly partial) function. Then f is said to be a *counter function* if

(i) $f(2) = 1$,

(ii) $f(n^2) = 2 \cdot f(n)$,

(iii) $f(n \cdot m) = f(n) + f(m)$.

2(b). Define $f_1 : M \to \mathbf{N}$ to be

$$f_1(2^k) = k.$$

Prove that f_1 is a counter function. (Hint: Induction on the definition of M.)

2(c). Prove that f_1 is the *unique* counter function, *i.e.*, if f_2 is any counter function, then $f_1(x) = f_2(x)$ for all $x \in M$.

2(d). Consider the function $g : M \to \mathbf{N}$ defined inductively by:

(i) $g(2) = 1$,

(ii) $g(n^2) = 5$,

(iii) $g(n \cdot m) = 10$.

Carefully prove that $1 = 0$. Explain why this contradiction occurs here, but not for counter functions.

(F)

2(a) Prove by induction on def of M that $M = \{2^k \mid k \geq 1\}$

ON K

(1) Basis $k=1 \rightarrow 2^k = 2^1 = 2$ ✓ by (i) $2 \in M$ ✓

(2) Induction: given $2^{k-1}, 2 \in M \rightarrow 2 \cdot 2^{k-1} = 2^k \in M$ by (iii) with $n = 2^{k-1}$, $m = 2$. ✓

This shows that all $\{2^k \mid k \geq 1\}$ is in M. ✓ I now show that all M is in $L = \{2^k \mid k \geq 1\}$ by considering the 3 cases.

(i) $2 \in M$ checks by def. ✓

inductive (ii) if $n \in M$, then $n^2 \in M$: $n \in L$ by ind then $n = 2^k$ for some $k \geq 1$. Thus $n^2 = 2^{2k}$ which implies $n^2 \in L$ ✓

inductive (iii) if $n, m \in M$, then $(n \cdot m) \in M$: $m, n \in L$ by ind thus $n = 2^k$ and $m = 2^j$ for some $j, k \geq 1$. Thus, $(n \cdot m) \in L$ since $2^k \cdot 2^j = 2^{k+j}$. ✓

Thus M consists exclusively of L and vice versa ($M = L$). □

(b) Prove f_1 is a counter function by inductive examination of def of a c.f.

(i) $f_1(2) = f_1(2^1) = 1$ ✓ ✓

(ii) $n = 2^k$ for some $k \geq 1$ $f_1(2^k) = k$ then $n^2 = 2^{2k}$ and $f_1(2^{2k}) = 2 \cdot k$.
Therefore $f_1(n^2) = f_1(2^{2k}) = 2 \cdot k = 2 \cdot f_1(2^k) = 2 \cdot f_1(n)$ ✓ ✓

(iii) $n = 2^k$ for some $k \geq 1 \rightarrow f_1(n) = f_1(2^k) = k$
$m = 2^j$ for some $j \geq 1$ $f_1(m) = f_1(2^j) = j$

$n \cdot m = 2^k \cdot 2^j = 2^{k+j} \rightarrow f_1(n \cdot m) = f_1(2^{k+j}) = k + j$

$f_1(n) + f_1(m) = k + j = f_1(n + m)$ ✓ ✓

25/25

2c. In part (a) we proved $M = \{ 2^k \mid k \geq 1 \}$

We prove by induction on $k = \lg x$ that ~~$\forall 2^k \in M.\ f_1$~~
$\forall x \in M.\ f_1(x) = f_2(x)$
in other words $\forall k \geq 1.\ f_1(2^k) = f_2(2^k)$.

Basis $k=1$. Then by def of f_1, $f_1(2^1) = 1$
Since f_2 is a counter function it must satisfy (i)
so ~~f~~ $f_2(2) = 1$
so $f_1(2^1) = f_2(2^1)$.

Ind. Step $k = k'+1$
Then by def of f_1, $f_1(2^k) = f_1(2^{k'+1}) = k'+1$

But since $2^k = 2^{k'+1} = 2 \cdot 2^{k'}$,
$f_2(2^k) = f_2(2 \cdot 2^{k'}) = f_2(2) + f_2(2^{k'})$ by ~~rule~~ condition (iii) on counter functions
by ~~rule~~ condition (i) on counter functions, $f_2(2) = 1$
by induction $f_2(2^{k'}) = k'$
so $f_2(2) + f_2(2^{k'}) = 1 + k'$
but $f_1(2^k) = k'+1$
so $f_2(2^k) = f_1(2^k)$ ∎

(d) The point of this problem was to show from the assumption that g was a function, that $10 = 5$ which is a contradiction. From a contradiction it is possible to prove any assertion at all!
But here are some specific steps:
since g is a function, $\forall x.\ g(x) = g(x)$
so ① $g(4) = g(4)$

	‖	‖
②	$g(2 \cdot 2)$	$g(2^2)$
	‖	‖
③	10	5

so by transitivity $10 = 5$.
divide both sides by 5 so $2 = 1$
subtract 1 from both sides, so $1 = 0$.

Problem 3. (The exercise on page 38 of Winskel) Give definitions of $loc(a)$, $loc(b)$, and $loc_R(c)$, the sets of locations which appear in arithmetic expressions a, boolean expressions b, and the right-hand sides of assignments in commands c.

Solution: We define $loc(a)$ the set of locations which appear in arithmetic expressions a as follows:

$$\begin{aligned} loc(n) &= \emptyset \\ loc(X) &= \{X\} \\ loc(a_0 + a_1) &= loc(a_0) \cup loc(a_1) \\ loc(a_0 \times a_1) &= loc(a_0) \cup loc(a_1) \\ loc(a_0 - a_1) &= loc(a_0) \cup loc(a_1) \end{aligned}$$

We define $loc(b)$ the set of locations which appear in boolean expressions b as follows:

$$\begin{aligned} loc(t) &= \emptyset \\ loc(a_0 = a_1) &= loc(a_0) \cup loc(a_1) \\ loc(a_0 \leq a_1) &= loc(a_0) \cup loc(a_1) \\ loc(b_0 \wedge b_1) &= loc(b_0) \cup loc(b_1) \\ loc(b_0 \vee b_1) &= loc(b_0) \cup loc(b_1) \\ loc(\neg b) &= loc(b) \end{aligned}$$

Notice that the definition for $loc(b)$ relied on the definition of $loc(a)$.

Finally, we take $loc_R(c)$ to literally be the set of locations which appear in the right-hand sides of assignments in c which gives:

$$\begin{aligned} loc_R(\mathbf{skip}) &= \emptyset \\ loc_R(X := a) &= loc(a) \\ loc_R(c_0; c_1) &= loc_R(c_0) \cup loc_R(c_1) \\ loc_R(\mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1) &= loc_R(c_0) \cup loc_R(c_1) \\ loc_R(\mathbf{while}\, b\, \mathbf{do}\, c') &= loc_R(c') \end{aligned}$$

Corrections

Quiz header

Quiz 1

Instructions. This is a **closed book** exam; no notes either. For your reference, there is an appendix listing the definitions of the "evaluates to" relation $\rightarrow$, and the one-step rewriting relation $\rightarrow_1$ on configurations of the language **IMP**.

Write your solutions for all four (4) problems on this exam sheet in the spaces provided, including your name on each sheet. Ask for further blank sheets if you need them. You may assume the results of previous parts in later parts of problems, so don't let "getting stuck" on any one part keep you from proceeding to later parts.

You have 50 mins. GOOD LUCK!

NAME ______________________

problem	*points*	*score*
1	(10)	
2	(15)	
3	(20)	
4	(25)	
Total	(70)	

NAME ______________________

For Problems 1 and 2, let w be the **IMP** command

$$\mathbf{while}\, 45 \leq X \,\mathbf{do}\, X := X - 3;\ Y := X - 1;\ X := Y - 1$$

and let σ be a state such that $\sigma(X) = 1000$ and $\sigma(Y) = 2000$.

Problem 1 [10 points]. According to the inductive definition of evaluation, the assertion

$$\langle w, \sigma \rangle \rightarrow \sigma[40/X][41/Y]$$

has a unique derivation. How many instances of the sequencing rule scheme (seq $\rightarrow$) given below appear in this derivation? [ ]

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'', \quad \langle c_1, \sigma'' \rangle \rightarrow \sigma'}{\langle (c_0; c_1), \sigma \rangle \rightarrow \sigma'} \qquad (\text{seq} \rightarrow)$$

Problem 2 [15 points]. By definition, $\langle w, \sigma \rangle \rightarrow_1^* \sigma[40/X][41/Y]$ because there is a (unique) sequence of the form:

$$\langle w, \sigma \rangle \rightarrow_1 \langle c_1, \sigma_1 \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle \rightarrow_1 \cdots \rightarrow_1 \langle c_n, \sigma_n \rangle \rightarrow_1 \sigma[40/X][41/Y]$$

where n happens to be 2500.

Notice that σ_1 must equal σ, and c_1 must be

$$\mathbf{if}\, 45 \leq X \,\mathbf{then}\, (X := X - 3;\ Y := X - 1;\ X := Y - 1;\ w) \,\mathbf{else}\, \mathbf{skip}.$$

Problem 2(a) [6 points]. What are

c_2? [ ]

σ_2? [ ]

c_3? [ ]

σ_3? [ ]

c_n? [ ]

σ_n? [ ]

- actually we're off by one, because $\langle w, \sigma \rangle$'s not $\langle c_0, \sigma_0 \rangle$

Problem 2(b) [4 points]. How many c_i's are of the form **while** b **do** c? [ ]

Problem 2(c) [5 points]. There are k times as many c_i's which are of the form **if** b' **then** c **else** c' than are of the form **while** b'' **do** c''. What is k? [ ]

Problem 3 [20 points]. It was noted in class that every **Aexp** configuration evaluates to a number. Likewise, one can prove by ***structural induction on*** **Bexp** that every **Bexp** configuration evaluates to a truth value, namely,

$$\text{for all } \langle b, \sigma \rangle, \text{ there is a } t \in \{\textbf{true}, \textbf{false}\} \text{ such that } \langle b, \sigma \rangle \rightarrow t.$$

Problem 3(a) [10 points]. List the cases of the structural induction and indicate what must be shown for each case.

Problem 3(b) [10 points]. Pick a non-base case and prove it!

Problem 4 [25 points]. We define a "parallel evals to" relation, $\hookrightarrow$, which is a variation of the "evals to" relation, $\rightarrow$. The rules to define $\hookrightarrow$ are obtained from the rules defining $\rightarrow$ by replacing all occurrences of "$\rightarrow$" by "$\hookrightarrow$". In addition, there is one further "parallel if" rule: Not entirely correct.

$$\frac{\langle c_0, \sigma\rangle \hookrightarrow \sigma', \quad \langle c_1, \sigma\rangle \hookrightarrow \sigma'}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \hookrightarrow \sigma'} \qquad (\text{par-if} \hookrightarrow)$$

Problem 4(a) [5 points]. Give a simple example of a command, c, such that $\langle c, \sigma\rangle \hookrightarrow \sigma$ has more than one derivation for any state σ.

Although the *definition* of $\hookrightarrow$ differs from that of $\rightarrow$, it turns out to specify the *same relation* on configurations as $\rightarrow$. The nontrivial direction of this remark is the implication

$$\langle c, \sigma\rangle \hookrightarrow \sigma' \text{ implies } \langle c, \sigma\rangle \rightarrow \sigma'.$$

This implication can be proved by induction on the definition of $\hookrightarrow$ (that is by rule induction on the rules for $\hookrightarrow$).

Problem 4(b) [10 points]. Briefly explain what the cases of the induction are, and why there is only one non-trivial case.

Problem 4(c) [10 points]. Prove the non-trivial case. (You may assume the results mentioned in Problem 3.)

name "eval" "elim"

A Appendix

We use n, sometimes with subscripts as in n_0, n_1, to denote arbitrary elements of **Num**. Similarly, we assume $X, Y \in$ **Loc**; $a \in$ **Aexp**, $t \in$ {**true**, **false**}; $b \in$ **Bexp**; $c \in$ **Com**; and $\sigma \in$ the set of states.

A.1 "Evals to" Rules for IMP

Notice that we give a name for each rule in parentheses to its right.

A.1.1 Aexp Rules

align labels

$$\langle n, \sigma \rangle \to n \qquad (\text{num} \to)$$

$$\langle X, \sigma \rangle \to \sigma(n) \qquad (\text{loc} \to)$$

$$\frac{\langle a_0, \sigma \rangle \to n_0, \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 + a_1, \sigma \rangle \to n} \qquad (\text{plus} \to)$$

where n is the sum of n_0 and n_1.

Similarly, there are rules (times $\to$) and (minus $\to$).

A.1.2 Bexp Rules

$$\frac{\langle a_0, \sigma \rangle \to n_0, \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 = a_1, \sigma \rangle \to t} \qquad (\text{equal} \to)$$

where $t \equiv$ **true** if n_0 and n_1 are equal, otherwise $t \equiv$ **false**.

Similarly, there is a rule ($\leq \to$).

$$\langle t, \sigma \rangle \to t \qquad (\text{bool} \to)$$

$$\frac{\langle b, \sigma \rangle \to t}{\langle \neg b, \sigma \rangle \to t'} \qquad (\text{not} \to)$$

where t' is the negation of t.

$$\frac{\langle b_0, \sigma \rangle \to t_0, \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \wedge b_1, \sigma \rangle \to t} \qquad (\text{and} \to)$$

where t is **true** if $t_0 \equiv$ **true** and $t_1 \equiv$ **true**, and is **false** otherwise.

Similarly, there is a rule (or $\to$).

A.1.3 Com Rules

$$\langle \mathbf{skip}, \sigma\rangle \to \sigma \qquad (\text{skip} \to)$$

$$\frac{\langle a, \sigma\rangle \to n}{\langle X := a, \sigma\rangle \to \sigma[n/X]} \qquad (\text{assign} \to)$$

$$\frac{\langle c_0, \sigma\rangle \to \sigma'', \quad \langle c_1, \sigma''\rangle \to \sigma'}{\langle (c_0; c_1), \sigma\rangle \to \sigma'} \qquad (\text{seq} \to)$$

$$\frac{\langle b, \sigma\rangle \to \mathbf{true}, \quad \langle c_0, \sigma\rangle \to \sigma'}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \to \sigma'} \qquad (\text{if-true} \to)$$

$$\frac{\langle b, \sigma\rangle \to \mathbf{false}, \quad \langle c_1, \sigma\rangle \to \sigma'}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \to \sigma'} \qquad (\text{if-false} \to)$$

$$\frac{\langle b, \sigma\rangle \to \mathbf{false}}{\langle \mathbf{while}\, b\, \mathbf{do}\, c, \sigma\rangle \to \sigma} \qquad (\text{while-false} \to)$$

$$\frac{\langle b, \sigma\rangle \to \mathbf{true}, \quad \langle c, \sigma\rangle \to \sigma'', \quad \langle \mathbf{while}\, b\, \mathbf{do}\, c, \sigma''\rangle \to \sigma'}{\langle \mathbf{while}\, b\, \mathbf{do}\, c, \sigma\rangle \to \sigma'} \qquad (\text{while-true} \to)$$

A.2 Rewriting rules for IMP

A.2.1 Aexp Rules

$$\langle X, \sigma\rangle \to_1 \langle \sigma(X), \sigma\rangle \qquad (\text{loc} \to_1)$$

$$\frac{\langle a_0, \sigma\rangle \to_1 \langle a_0', \sigma\rangle}{\langle a_0 + a_1, \sigma\rangle \to_1 \langle a_0' + a_1, \sigma\rangle} \qquad (\text{plus-left} \to_1)$$

$$\frac{\langle a, \sigma\rangle \to_1 \langle a', \sigma\rangle}{\langle n + a, \sigma\rangle \to_1 \langle n + a', \sigma\rangle} \qquad (\text{plus-right} \to_1)$$

$$\langle n_0 + n_1, \sigma\rangle \to_1 \langle n, \sigma\rangle \qquad (\text{plus-num} \to_1)$$

where n is the sum of n_0 and n_1.

Similarly, there are rules (times-left $\to_1$), (times-right $\to_1$), (times-num $\to_1$), (minus-left $\to_1$), (minus-right $\to_1$), and (minus-num $\to_1$).

A.2.2 Bexp Rules

$$\frac{\langle a_0, \sigma\rangle \rightarrow_1 \langle a_0', \sigma\rangle}{\langle a_0 = a_1, \sigma\rangle \rightarrow_1 \langle a_0' = a_1, \sigma\rangle} \qquad \text{(equal-left} \rightarrow_1\text{)}$$

$$\frac{\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle}{\langle n = a, \sigma\rangle \rightarrow_1 \langle n = a', \sigma\rangle} \qquad \text{(equal-right} \rightarrow_1\text{)}$$

$$\langle n_0 = n_1, \sigma\rangle \rightarrow_1 \langle t, \sigma\rangle \qquad \text{(equal-num} \rightarrow_1\text{)}$$

where $t \equiv$ **true** if n_0 and n_1 are equal, otherwise $t \equiv$ **false**.

Similarly, there are rules ($\leq$-left $\rightarrow_1$), ($\leq$-right $\rightarrow_1$), and ($\leq$-num $\rightarrow_1$).

$$\frac{\langle b, \sigma\rangle \rightarrow_1 \langle b', \sigma\rangle}{\langle \neg b, \sigma\rangle \rightarrow_1 \langle \neg b', \sigma\rangle} \qquad \text{(not-eval-arg} \rightarrow_1\text{)}$$

$$\langle \neg t, \sigma\rangle \rightarrow_1 \langle t', \sigma\rangle \qquad \text{(not-bool} \rightarrow_1\text{)}$$

where t' is the negation of t.

$$\frac{\langle b_0, \sigma\rangle \rightarrow_1 \langle b_0', \sigma\rangle}{\langle b_0 \wedge b_1, \sigma\rangle \rightarrow_1 \langle b_0' \wedge b_1, \sigma\rangle} \qquad \text{(and-left} \rightarrow_1\text{)}$$

$$\frac{\langle b, \sigma\rangle \rightarrow_1 \langle b', \sigma\rangle}{\langle t \wedge b, \sigma\rangle \rightarrow_1 \langle t \wedge b', \sigma\rangle} \qquad \text{(and-right} \rightarrow_1\text{)}$$

$$\langle t_0 \wedge t_1, \sigma\rangle \rightarrow_1 \langle t, \sigma\rangle \qquad \text{(and-bool} \rightarrow_1\text{)}$$

where $t \equiv$ **true** if $t_0 \equiv$ **true** and $t_1 \equiv$ **true**, otherwise $t \equiv$ **false**.

Similarly there are rules (or-left $\rightarrow_1$), (or-right, $\rightarrow_1$) and (or-bool $\rightarrow_1$).

A.2.3 Com Rules

$$\langle \textbf{skip}, \sigma\rangle \rightarrow_1 \sigma \qquad \text{(skip} \rightarrow_1\text{)}$$

$$\frac{\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle}{\langle X := a, \sigma\rangle \rightarrow_1 \langle X := a', \sigma\rangle} \qquad \text{(assign-eval-arg} \rightarrow_1\text{)}$$

$$\langle X := n, \sigma\rangle \rightarrow_1 \sigma[n/X] \qquad \text{(assign-num} \rightarrow_1\text{)}$$

$$\frac{\langle c_0, \sigma\rangle \to_1 \langle c_0', \sigma'\rangle}{\langle (c_0; c_1), \sigma\rangle \to_1 \langle (c_0'; c_1), \sigma'\rangle} \qquad \text{(seq-start} \to_1\text{)}$$

$$\frac{\langle c_0, \sigma\rangle \to_1 \sigma'}{\langle (c_0; c_1), \sigma\rangle \to_1 \langle c_1, \sigma'\rangle} \qquad \text{(seq-finish} \to_1\text{)}$$

$$\frac{\langle b, \sigma\rangle \to_1 \langle b', \sigma\rangle}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \to_1 \langle \mathbf{if}\, b'\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle} \qquad \text{(if-eval-guard} \to_1\text{)}$$

$$\langle \mathbf{if\, true\, then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \to_1 \langle c_0, \sigma\rangle \qquad \text{(if-true} \to_1\text{)}$$

$$\langle \mathbf{if\, false\, then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \to_1 \langle c_1, \sigma\rangle \qquad \text{(if-false} \to_1\text{)}$$

$$\langle \mathbf{while}\, b\, \mathbf{do}\, c, \sigma\rangle \to_1 \langle \mathbf{if}\, b\, \mathbf{then} (c; \mathbf{while}\, b\, \mathbf{do}\, c)\, \mathbf{else\, skip}, \sigma\rangle \qquad \text{(while} \to_1\text{)}$$

6.044J/18.423J: Computability, Programming, and Logic — Handout 17
Massachusetts Institute of Technology — 9 October 1991

Quiz 1 Solutions

Instructions. This was a **closed book** exam; no notes either. For your reference, there is an appendix listing the definitions of the "evaluates to" relation $\rightarrow$, and the one-step rewriting relation $\rightarrow_1$ on configurations of the language **IMP**.

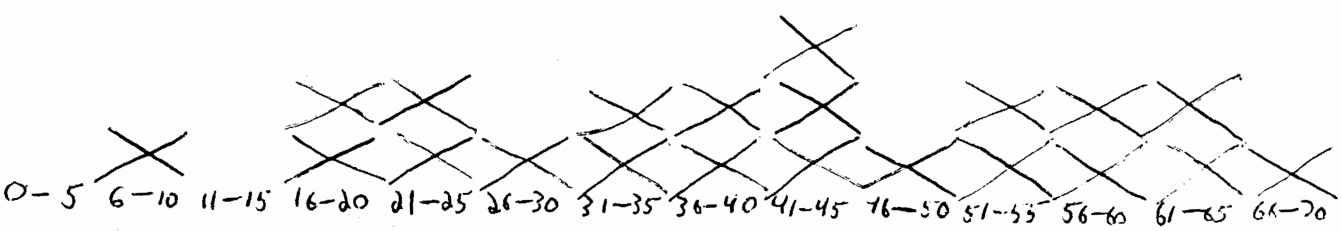
Write your solutions for all four (4) problems on this exam sheet in the spaces provided, including your name on each sheet. Ask for further blank sheets if you need them. You may assume the results of previous parts in later parts of problems, so don't let "getting stuck" on any one part keep you from proceeding to later parts.

The exam was 50 minutes.

The exam was graded out of a possible total of 70 points. The point values are indicated on each problem. The overall statistics are as follows:

Number Submitted	21
High	66
Low	6
Median	41
Mean	40.8

The following is a histogram of the grade distribution:



For Problems 1 and 2, let w be the **IMP** command

$$\textbf{while}\, 45 \leq X \,\textbf{do}\, X := X - 3;\ Y := X - 1;\ X := Y - 1$$

and let σ be a state such that $\sigma(X) = 1000$ and $\sigma(Y) = 2000$.

Problem 1 [10 points]. According to the inductive definition of evaluation, the assertion

$$\langle w, \sigma\rangle \rightarrow \sigma[40/X][41/Y]$$

has a unique derivation. How many instances of the sequencing rule scheme (seq →) given below appear in this derivation? | *384* |

$$\frac{\langle c_0, \sigma\rangle \rightarrow \sigma'', \quad \langle c_1, \sigma''\rangle \rightarrow \sigma'}{\langle (c_0; c_1), \sigma\rangle \rightarrow \sigma'} \qquad (\text{seq} \rightarrow)$$

Note: The quiz did not ask for any explanation. One will be asked for on problem set 4.

Problem 2 [15 points]. By definition, $\langle w, \sigma\rangle \rightarrow_1^* \sigma[40/X][41/Y]$ because there is a (unique) sequence of the form:

$$\langle w, \sigma\rangle \rightarrow_1 \langle c_1, \sigma_1\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle \rightarrow_1 \cdots \rightarrow_1 \langle c_n, \sigma_n\rangle \rightarrow_1 \sigma[40/X][41/Y]$$

where n happens to be 2500.

Notice that σ_1 must equal σ, and c_1 must be

$$\textbf{if}\, 45 \leq X \,\textbf{then}\, (X := X - 3;\ Y := X - 1;\ X := Y - 1;\ w) \,\textbf{else}\, \textbf{skip}.$$

Problem 2(a) [6 points]. What are

c_2? | $\textbf{if}\, 45 \leq 1000 \,\textbf{then}\, (X := X - 3;\ Y := X - 1;\ X := Y - 1;\ w) \,\textbf{else}\, \textbf{skip}$ |

σ_2? | σ |

c_3? | $\textbf{if}\, \textbf{true} \,\textbf{then}\, (X := X - 3;\ Y := X - 1;\ X := Y - 1;\ w) \,\textbf{else}\, \textbf{skip}$ |

σ_3? | σ

c_n? | **skip**

σ_n? | $\sigma[40/X][41/Y]$

The answers for c_3 and σ_3 were graded relative to the answers for c_2 and σ_2.

Problem 2(b) [4 points]. How many c_i's are of the form **while** b **do** c? | *193*

Actually the correct answer is really 192. We forgot that the first configuration in the chain ($\langle w, \sigma \rangle$) was not explicitly described to be c_0. Thus the first **while** *in the chain does not really contribute to the count. If we had let $\langle c_0, \sigma_0 \rangle = \langle w, \sigma \rangle$ then there would not have been a problem. Full credit was given for either answer, unless it was clear that 192 was arrived at via a mistake (and thus two wrongs making a right).*

Problem 2(c) [5 points]. There are k times as many c_i's which are of the form **if** b' **then** c **else** c' than are of the form **while** b'' **do** c''. What is k? | *3*

Due to the slight miscounting in the preceding answer the correct answer is really $(193 * 3)/192 \approx 3.0156$, *credit was given for either answer.*

Problem 3 [20 points]. It was noted in class that every **Aexp** configuration evaluates to a number. Likewise, one can prove by *structural induction on* **Bexp** that every **Bexp** configuration evaluates to a truth value, namely,

$$\text{for all } \langle b, \sigma \rangle, \text{ there is a } t \in \{\textbf{true}, \textbf{false}\} \text{ such that } \langle b, \sigma \rangle \rightarrow t.$$

Problem 3(a) [10 points]. List the cases of the structural induction and indicate what must be shown for each case.

The base cases are:

[$b \equiv t \in \{\textbf{true}, \textbf{false}\}$] *We must show that, under no additional assumptions, there exists a $t' \in \{\textbf{true}, \textbf{false}\}$ such that $\langle t, \sigma \rangle \rightarrow t'$.*

[$b \equiv a_0 = a_1$] *We must show that there exists a* $t \in \{\mathbf{true}, \mathbf{false}\}$ *such that* $\langle a_0 = a_1, \sigma\rangle \rightarrow t$. *To do so, we may use the analogous property for* **Aexp**, *viz. to assume that there exist* n_0 *and* n_1 *such that* $\langle a_0, \sigma\rangle \rightarrow n_0$ *and* $\langle a_1, \sigma\rangle \rightarrow n_1$.

[$b \equiv a_0 \leq a_1$] *Similar to the preceding case.*

The non-base cases are:

[$b \equiv \neg b'$] *Under the assumption that there exists a* $t \in \{\mathbf{true}, \mathbf{false}\}$ *such that* $\langle b', \sigma\rangle \rightarrow t$, *we must show that there exists a* $t' \in \{\mathbf{true}, \mathbf{false}\}$ *such that* $\langle \neg b', \sigma\rangle \rightarrow t'$.

[$b \equiv b_0 \wedge b_1$] *Under the assumption that there exist* $t_0, t_1 \in \{\mathbf{true}, \mathbf{false}\}$ *such that* $\langle b_0, \sigma\rangle \rightarrow t_0$ *and* $\langle b_1, \sigma\rangle \rightarrow t_1$, *we must show that there exists a* $t \in \{\mathbf{true}, \mathbf{false}\}$ *such that* $\langle b_0 \wedge b_1, \sigma\rangle \rightarrow t$.

[$b \equiv b_0 \vee b_1$] *Similar to the preceding case.*

Problem 3(b) [10 points]. Pick a non-base case and prove it!

We pick the non base-case [$b \equiv b_0 \wedge b_1$].

Under the inductive assumption that there exist $t_0, t_1 \in \{\mathbf{true}, \mathbf{false}\}$ *such that* $\langle b_0, \sigma\rangle \rightarrow t_0$ *and* $\langle b_1, \sigma\rangle \rightarrow t_1$, *we must show that there exists a* $t \in \{\mathbf{true}, \mathbf{false}\}$ *such that* $\langle b_0 \wedge b_1, \sigma\rangle \rightarrow t$. *But by rule (and* $\rightarrow$*), there is such a* t, *namely the conjunction of* t_0 *and* t_1.

Problem 4 [25 points]. We define a "parallel evals to" relation, $\hookrightarrow$, which is a variation of the "evals to" relation, $\rightarrow$. The rules to define $\hookrightarrow$ are obtained from the rules defining $\rightarrow$ by replacing all occurrences of "$\rightarrow$" by "$\hookrightarrow$". In addition, there is one further "parallel if" rule:

$$\frac{\langle c_0, \sigma\rangle \hookrightarrow \sigma', \quad \langle c_1, \sigma\rangle \hookrightarrow \sigma'}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \hookrightarrow \sigma'} \qquad (\text{par-if} \hookrightarrow)$$

Problem 4(a) [5 points]. Give a simple example of a command, c, such that $\langle c, \sigma\rangle \hookrightarrow \sigma$ has more than one derivation for any state σ.

$\mathbf{if}\, b\, \mathbf{then}\, c'\, \mathbf{else}\, c'$, *for any* c'

Although the *definition* of $\hookrightarrow$ differs from that of $\rightarrow$, it turns out to specify the *same relation* on configurations as $\rightarrow$. The nontrivial direction of this remark is the implication

$$\langle c, \sigma\rangle \hookrightarrow \sigma' \text{ implies } \langle c, \sigma\rangle \rightarrow \sigma'.$$

This implication can be proved by induction on the definition of $\hookrightarrow$ (that is by rule induction on the rules for $\hookrightarrow$).

Problem 4(b) [10 points]. Briefly explain what the cases of the induction are, and why there is only one non-trivial case.

There is one case for each of the inference rules of $\hookrightarrow$ on **Com**-*configurations (or on* **Aexp**, **Bexp**, *and* **Com** *configurations. This was slightly ambiguous but unimportant, since either reading gave the same definition of $\hookrightarrow$).*

So there are the base cases for the **Com**-*configuration rules: (skip $\hookrightarrow$), (assign $\hookrightarrow$).*

The inductive cases are for the rules: (seq $\hookrightarrow$), (if-true $\hookrightarrow$), (if-false $\hookrightarrow$) (while-false $\hookrightarrow$), (while-false $\hookrightarrow$), and finally a case for the new rule (par-if $\hookrightarrow$).

The only non-trivial case is for the new rule (par-if $\hookrightarrow$), because the other rules for $\hookrightarrow$ are the same as the corresponding rules for $\rightarrow$. In particular, if $\langle c, \sigma \rangle \hookrightarrow \sigma'$ follows from some ($\hookrightarrow$)-rule, R, other than (par-if $\rightarrow$), then the antecedents if any, of R which involve $\hookrightarrow$, each implies by induction, the corresponding antecedent with "$\hookrightarrow$" replaced by "$\rightarrow$", so $\langle c, \sigma \rangle \rightarrow \sigma'$ follows trivially by the $\rightarrow$-version of R.

Problem 4(c) [10 points]. Prove the non-trivial case. (You may assume the results mentioned in Problem 3.)

So, we suppose that $\langle c, \sigma \rangle \hookrightarrow \sigma'$ because of the rule (par-if $\hookrightarrow$).

Then c must be of the form $\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1$, *where $\langle c_0, \sigma \rangle \hookrightarrow \sigma'$ and $\langle c_1, \sigma \rangle \hookrightarrow \sigma'$ (so by induction, we may assume that $\langle c_0, \sigma \rangle \rightarrow \sigma'$ and $\langle c_1, \sigma \rangle \rightarrow \sigma'$).*

By Problem 3, we know that there exists a $t \in \{\mathbf{true}, \mathbf{false}\}$ such that $\langle b, \sigma \rangle \rightarrow t$. This gives us two cases based on t.

Suppose $t \equiv$ **true**. *Since $\langle b, \sigma \rangle \rightarrow \mathbf{true}$, rule (if-true $\rightarrow$) applies, and so then $\langle c, \sigma \rangle \rightarrow \sigma'$.*

The case of $t \equiv$ **false** *works similarly. Since $\langle b, \sigma \rangle \rightarrow \mathbf{false}$, rule (if-false $\rightarrow$) applies, and so then $\langle c, \sigma \rangle \rightarrow \sigma'$.*

A Appendix

We use n, sometimes with subscripts as in n_0, n_1, to denote arbitrary elements of **Num**. Similarly, we assume $X, Y \in$ **Loc**; $a \in$ **Aexp**, $t \in \{\textbf{true}, \textbf{false}\}$; $b \in$ **Bexp**; $c \in$ **Com**; and $\sigma \in$ the set of states.

A.1 "Evals to" Rules for IMP

Notice that we give a name for each rule in parentheses to its right.

A.1.1 Aexp Rules

$$\langle n, \sigma \rangle \rightarrow n \qquad \text{(num} \rightarrow\text{)}$$

$$\langle X, \sigma \rangle \rightarrow \sigma(n) \qquad \text{(loc} \rightarrow\text{)}$$

$$\frac{\langle a_0, \sigma \rangle \rightarrow n_0, \quad \langle a_1, \sigma \rangle \rightarrow n_1}{\langle a_0 + a_1, \sigma \rangle \rightarrow n} \qquad \text{(plus} \rightarrow\text{)}$$

where n is the sum of n_0 and n_1.

Similarly, there are rules (times $\rightarrow$) and (minus $\rightarrow$).

A.1.2 Bexp Rules

$$\langle t, \sigma \rangle \rightarrow t \qquad \text{(bool} \rightarrow\text{)}$$

$$\frac{\langle a_0, \sigma \rangle \rightarrow n_0, \quad \langle a_1, \sigma \rangle \rightarrow n_1}{\langle a_0 = a_1, \sigma \rangle \rightarrow t} \qquad \text{(equal} \rightarrow\text{)}$$

where $t \equiv$ **true** if n_0 and n_1 are equal, otherwise $t \equiv$ **false**.

Similarly, there is a rule ($\leq \rightarrow$).

$$\frac{\langle b, \sigma \rangle \rightarrow t}{\langle \neg b, \sigma \rangle \rightarrow t'} \qquad \text{(not} \rightarrow\text{)}$$

where t' is the negation of t.

$$\frac{\langle b_0, \sigma \rangle \rightarrow t_0, \quad \langle b_1, \sigma \rangle \rightarrow t_1}{\langle b_0 \wedge b_1, \sigma \rangle \rightarrow t} \qquad \text{(and} \rightarrow\text{)}$$

where t is **true** if $t_0 \equiv$ **true** and $t_1 \equiv$ **true**, and is **false** otherwise.

Similarly, there is a rule (or $\rightarrow$).

A.1.3 Com Rules

$$\langle \textbf{skip}, \sigma\rangle \to \sigma \qquad (\text{skip} \to)$$

$$\frac{\langle a, \sigma\rangle \to n}{\langle X := a, \sigma\rangle \to \sigma[n/X]} \qquad (\text{assign} \to)$$

$$\frac{\langle c_0, \sigma\rangle \to \sigma'', \quad \langle c_1, \sigma''\rangle \to \sigma'}{\langle (c_0; c_1), \sigma\rangle \to \sigma'} \qquad (\text{seq} \to)$$

$$\frac{\langle b, \sigma\rangle \to \textbf{true}, \quad \langle c_0, \sigma\rangle \to \sigma'}{\langle \textbf{if}\, b\, \textbf{then}\, c_0\, \textbf{else}\, c_1, \sigma\rangle \to \sigma'} \qquad (\text{if-true} \to)$$

$$\frac{\langle b, \sigma\rangle \to \textbf{false}, \quad \langle c_1, \sigma\rangle \to \sigma'}{\langle \textbf{if}\, b\, \textbf{then}\, c_0\, \textbf{else}\, c_1, \sigma\rangle \to \sigma'} \qquad (\text{if-false} \to)$$

$$\frac{\langle b, \sigma\rangle \to \textbf{false}}{\langle \textbf{while}\, b\, \textbf{do}\, c, \sigma\rangle \to \sigma} \qquad (\text{while-false} \to)$$

$$\frac{\langle b, \sigma\rangle \to \textbf{true}, \quad \langle c, \sigma\rangle \to \sigma'', \quad \langle \textbf{while}\, b\, \textbf{do}\, c, \sigma''\rangle \to \sigma'}{\langle \textbf{while}\, b\, \textbf{do}\, c, \sigma\rangle \to \sigma'} \qquad (\text{while-true} \to)$$

A.2 Rewriting rules for IMP

A.2.1 Aexp Rules

$$\langle X, \sigma\rangle \to_1 \langle \sigma(X), \sigma\rangle \qquad (\text{loc} \to_1)$$

$$\frac{\langle a_0, \sigma\rangle \to_1 \langle a_0', \sigma\rangle}{\langle a_0 + a_1, \sigma\rangle \to_1 \langle a_0' + a_1, \sigma\rangle} \qquad (\text{plus-left} \to_1)$$

$$\frac{\langle a, \sigma\rangle \to_1 \langle a', \sigma\rangle}{\langle n + a, \sigma\rangle \to_1 \langle n + a', \sigma\rangle} \qquad (\text{plus-right} \to_1)$$

$$\langle n_0 + n_1, \sigma\rangle \to_1 \langle n, \sigma\rangle \qquad (\text{plus-num} \to_1)$$

where n is the sum of n_0 and n_1.

Similarly, there are rules (times-left $\to_1$), (times-right $\to_1$), (times-num $\to_1$), (minus-left $\to_1$), (minus-right $\to_1$), and (minus-num $\to_1$).

A.2.2 Bexp Rules

$$\frac{\langle a_0, \sigma\rangle \rightarrow_1 \langle a_0', \sigma\rangle}{\langle a_0 = a_1, \sigma\rangle \rightarrow_1 \langle a_0' = a_1, \sigma\rangle} \qquad \text{(equal-left } \rightarrow_1)$$

$$\frac{\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle}{\langle n = a, \sigma\rangle \rightarrow_1 \langle n = a', \sigma\rangle} \qquad \text{(equal-right } \rightarrow_1)$$

$$\langle n_0 = n_1, \sigma\rangle \rightarrow_1 \langle t, \sigma\rangle \qquad \text{(equal-num } \rightarrow_1)$$

where $t \equiv$ **true** if n_0 and n_1 are equal, otherwise $t \equiv$ **false**.

Similarly, there are rules ($\leq$-left $\rightarrow_1$), ($\leq$-right $\rightarrow_1$), and ($\leq$-num $\rightarrow_1$).

$$\frac{\langle b, \sigma\rangle \rightarrow_1 \langle b', \sigma\rangle}{\langle \neg b, \sigma\rangle \rightarrow_1 \langle \neg b', \sigma\rangle} \qquad \text{(not-eval-arg } \rightarrow_1)$$

$$\langle \neg t, \sigma\rangle \rightarrow_1 \langle t', \sigma\rangle \qquad \text{(not-bool } \rightarrow_1)$$

where t' is the negation of t.

$$\frac{\langle b_0, \sigma\rangle \rightarrow_1 \langle b_0', \sigma\rangle}{\langle b_0 \wedge b_1, \sigma\rangle \rightarrow_1 \langle b_0' \wedge b_1, \sigma\rangle} \qquad \text{(and-left } \rightarrow_1)$$

$$\frac{\langle b, \sigma\rangle \rightarrow_1 \langle b', \sigma\rangle}{\langle t \wedge b, \sigma\rangle \rightarrow_1 \langle t \wedge b', \sigma\rangle} \qquad \text{(and-right } \rightarrow_1)$$

$$\langle t_0 \wedge t_1, \sigma\rangle \rightarrow_1 \langle t, \sigma\rangle \qquad \text{(and-bool } \rightarrow_1)$$

where $t \equiv$ **true** if $t_0 \equiv$ **true** and $t_1 \equiv$ **true**, otherwise $t \equiv$ **false**.

Similarly there are rules (or-left $\rightarrow_1$), (or-right, $\rightarrow_1$) and (or-bool $\rightarrow_1$).

A.2.3 Com Rules

$$\langle \mathbf{skip}, \sigma\rangle \rightarrow_1 \sigma \qquad \text{(skip } \rightarrow_1)$$

$$\frac{\langle a, \sigma\rangle \rightarrow_1 \langle a', \sigma\rangle}{\langle X := a, \sigma\rangle \rightarrow_1 \langle X := a', \sigma\rangle} \qquad \text{(assign-eval-arg } \rightarrow_1)$$

$$\langle X := n, \sigma\rangle \rightarrow_1 \sigma[n/X] \qquad \text{(assign-num } \rightarrow_1)$$

$$\frac{\langle c_0, \sigma\rangle \rightarrow_1 \langle c_0', \sigma'\rangle}{\langle (c_0; c_1), \sigma\rangle \rightarrow_1 \langle (c_0'; c_1), \sigma'\rangle} \qquad (\textbf{seq-start} \rightarrow_1)$$

$$\frac{\langle c_0, \sigma\rangle \rightarrow_1 \sigma'}{\langle (c_0; c_1), \sigma\rangle \rightarrow_1 \langle c_1, \sigma'\rangle} \qquad (\textbf{seq-finish} \rightarrow_1)$$

$$\frac{\langle b, \sigma\rangle \rightarrow_1 \langle b', \sigma\rangle}{\langle \textbf{if}\, b\, \textbf{then}\, c_0\, \textbf{else}\, c_1, \sigma\rangle \rightarrow_1 \langle \textbf{if}\, b'\, \textbf{then}\, c_0\, \textbf{else}\, c_1, \sigma\rangle} \qquad (\text{if-eval-guard} \rightarrow_1)$$

$$\langle \textbf{if}\, \textbf{true}\, \textbf{then}\, c_0\, \textbf{else}\, c_1, \sigma\rangle \rightarrow_1 \langle c_0, \sigma\rangle \qquad (\text{if-true} \rightarrow_1)$$

$$\langle \textbf{if}\, \textbf{false}\, \textbf{then}\, c_0\, \textbf{else}\, c_1, \sigma\rangle \rightarrow_1 \langle c_1, \sigma\rangle \qquad (\text{if-false} \rightarrow_1)$$

$$\langle \textbf{while}\, b\, \textbf{do}\, c, \sigma\rangle \rightarrow_1 \langle \textbf{if}\, b\, \textbf{then}(c; \textbf{while}\, b\, \textbf{do}\, c)\, \textbf{else}\, \textbf{skip}, \sigma\rangle \qquad (\text{while} \rightarrow_1)$$

Problem Set 4

Due: 18 October 1991

Reading for the Problem Set. Winskel through §5.3.

Reading for Lectures. Winskel through §5.4.

Problem 1. Prove the claim made in Winskel, §5.2, p.54, l.-10 that $\Gamma = \hat{R}$.

In Problem Set 1 we looked at extending **IMP** with a construct **resultis** which made side effects uniformly available within all **Aexp**'s. Call this language $\mathbf{IMP}_r$. We now consider a very similar extension of **IMP** called $\mathbf{IMP}_v$ obtained by adding "**valis**" construct to **IMP**. Like **resultis**, the purpose of **valis** is to allow **Aexp**'s to have commands within them, but unlike **resultis**, the **valis** construct will preserve the property of **IMP** that there are no net side-effects in evaluating **Aexp**'s and **Bexp**'s.

The BNF grammar for $\mathbf{IMP}_v$ is exactly like the grammar for **IMP** except for the case of **Aexp**, which is now:

$$a ::= n \mid X \mid c\,\mathbf{valis}\,a \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

We distinguish the expressions, evaluation relations, *etc.*, for **IMP**, $\mathbf{IMP}_r$, and $\mathbf{IMP}_v$, with corresponding subscripts, *e.g.*, $\rightarrow_r$ for $\mathbf{IMP}_r$ evaluation and $\mathbf{Aexp}_v$ for the **Aexp**'s of $\mathbf{IMP}_v$.

The rules defining $\rightarrow_v$ include all of the rules for defining $\rightarrow$ for **IMP** (with "$\rightarrow$" changed to "$\rightarrow_v$") for **Aexp**'s and **Bexp**'s and **Com**'s. In addition, we add one further rule for **valis**:

$$\frac{\langle c, \sigma\rangle \rightarrow_v \sigma', \quad \langle a, \sigma'\rangle \rightarrow_v n}{\langle c\,\mathbf{valis}\,a, \sigma\rangle \rightarrow_v n} \qquad (\text{valis} \rightarrow_v)$$

To get an intuition for the difference between $\mathbf{IMP}_r$ and $\mathbf{IMP}_v$, consider the intended behavior of the **Aexp**'s

$$\begin{aligned} a_r &= \text{Euclid}\,\mathbf{resultis}\,M, \\ a_v &= \text{Euclid}\,\mathbf{valis}\,M. \end{aligned}$$

The behavior of a_r in state σ is to evaluate Euclid until it stops in some state σ'. It returns *the configuration* $\langle\sigma'(M), \sigma'\rangle$ so further evaluation will continue from state σ'. So this $\mathbf{Aexp}_r$ has some nasty side-effects—in computing the gcd of M and N, it has usually changed the values stored in those locations!

The behavior of a_v in a state σ is to evaluate Euclid in σ until it stops in some state σ'. It returns $\sigma'(M) \in \mathbf{Num}$. Further evaluation will continue from the original state σ as in **IMP**—the side-effected state σ' is discarded. $\mathbf{Aexp}_v$'s can be easier to understand and use: this one computes the gcd of M and N *without* affecting the values stored in those locations! (On the other hand, $\mathbf{IMP}_v$ may be more complicated to implement than $\mathbf{IMP}_r$—consider how you might define $\rightarrow_{1,v}$.)

The next problem is designed to further highlight the difference between $\mathbf{IMP}_r$ and $\mathbf{IMP}_v$. For example, addition is commutative in $\mathbf{IMP}_v$ but not in $\mathbf{IMP}_r$.

Problem 2.

2(a). Exhibit $a_0, a_1 \in \mathbf{Aexp}_r, n_0 \neq n_1 \in \mathbf{Num}, \sigma, \sigma' \in \Sigma$ such that

$$\langle a_0 + a_1, \sigma\rangle \rightarrow_r \langle n_0, \sigma'\rangle$$

and

$$\langle a_1 + a_0, \sigma\rangle \rightarrow_r \langle n_1, \sigma'\rangle.$$

(Addition is not commutative in $\mathbf{IMP}_r$ in that $n_0 \neq n_1$).

2(b). Outline a proof that for all $a_0, a_1 \in \mathbf{Aexp}_v, n \in \mathbf{Num}, \sigma \in \Sigma$ that

$$\langle a_0 + a_1, \sigma\rangle \rightarrow_v n \text{ iff } \langle a_1 + a_0, \sigma\rangle \rightarrow_v n.$$

(So addition is commutative in $\mathbf{IMP}_v$.)

Problem 3. Let $\hookrightarrow_v$ be defined by adding the (par-if) rule to the rules for $\rightarrow_v$ as done on Quiz 1 for **IMP**.

$$\frac{\langle c_0, \sigma\rangle \hookrightarrow_v \sigma', \quad \langle c_1, \sigma\rangle \hookrightarrow_v \sigma'}{\langle \mathbf{if}\; b\; \mathbf{then}\; c_0\; \mathbf{else}\; c_1, \sigma\rangle \hookrightarrow_v \sigma'} \qquad (\text{par-if} \hookrightarrow_v)$$

3(a). Briefly sketch how to prove that $\langle c, \sigma\rangle \rightarrow_v \sigma'$ implies $\langle c, \sigma\rangle \hookrightarrow_v \sigma'$.

3(b). Give a simple $\mathbf{IMP}_v$ command configuration which is a counterexample to the claim that $\hookrightarrow_v$ implies $\rightarrow_v$. Briefly explain where the proof on Quiz 1 of the corresponding implication for **IMP** breaks down for $\mathbf{IMP}_v$.

It's worth remarking that commands and expressions of **IMP** are a special case of those of $\mathbf{IMP}_v$. The ambiguity is harmless because, **IMP** commands evaluate the same using the **IMP** or the $\mathbf{IMP}_v$ evaluator ($\mathbf{IMP}_v$ might be called a *faithful* extension of **IMP**). More formally, for all commands c of **IMP**:

$$\langle c, \sigma \rangle \rightarrow \sigma' \text{ iff } \langle c, \sigma \rangle \rightarrow_v \sigma'.$$

The proof is a trivial rule induction, and we omit it. This also holds for $\mathbf{IMP}_r$; it also holds with $\hookrightarrow$ replacing $\rightarrow$ (on both sides of the "iff").

Problem 4.

4(a). Give the definition of a denotational semantics for $\mathbf{IMP}_v$ by structural induction. (Your definition should satisfy the result of problem 4(b).)

4(b). The proof of the equivalence of the operational and denotational semantics for **IMP** in Winskel §5.3 carries over to $\mathbf{IMP}_v$ with only minor changes. Briefly, but clearly, indicate the changes needed in the proof in §5.3 to prove that for all $c \in \mathbf{Com}_v$,

$$\langle c, \sigma \rangle \rightarrow_v \sigma' \text{ iff } \mathcal{C}[\![c]\!](\sigma) = \sigma'.$$

Revised Problem Set 5

Reading for the Problem Set. Winskel through §5.4.

Reading for Lectures. Winskel Chapter 6.

Due: 25 October 1991

We define an extension, $\mathbf{IMP}_{let}$, of the language **IMP**, by adding a recursive command declaration construct **letrec**. Expressions of $\mathbf{IMP}_{let}$ are exactly those of **IMP**. Commands, $\mathbf{Com}_{let}$, of $\mathbf{IMP}_{let}$ are those of **IMP**, with an additional **letrec** construct. Let p range over a new class, **OCom**, of *Open Commands*. Open commands are simply commands in which undeclared "procedure identifiers" $\mathbf{p}_i$ for $i \geq 1$ may appear.

$\mathbf{Com}_{let}$ and **OCom** are specified by the following grammar, subject to a "well-formedness" condition on **letrec** described below.

$$c ::= \mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, p' \mid \mathbf{skip} \mid X := a \mid c_0; c_1 \mid \mathbf{if}\, b \,\mathbf{then}\, c_0 \,\mathbf{else}\, c_1 \mid \mathbf{while}\, b \,\mathbf{do}\, c$$

$$p ::= \mathbf{p}_i \mid c \mid p_0; p_1 \mid \mathbf{if}\, b \,\mathbf{then}\, p_0 \,\mathbf{else}\, p_1 \mid \mathbf{while}\, b \,\mathbf{do}\, p$$

An expression of the form $\mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, p'$ where $\vec{p}_n = p_1, \ldots, p_n$ for $n \geq 1$, is well-formed only if all $\mathbf{p}_i$ occurring in the "declaration bodies" $p_1, \ldots, p_n$ and the "procedure body" p' are such that $i \leq n$. The idea is that $\mathbf{p}_i$ is bound to p_i for $1 \leq i \leq n$. A **letrec** binds all of the $\mathbf{p}_i$'s occurring anywhere within it, so $\mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, p'$ is a (closed) command.

It is important to note that the $\mathbf{p}_i$ are *not* metavariables ranging over different identifiers, but specific identifiers which are reused and bound to different "procedure bodies" p_i by different $(\mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, \cdot)$ constructs.

Evaluation rules of $\mathbf{IMP}_{let}$ for **Aexp** and **Bexp** are the same as in **IMP**.

The rules for $\mathbf{Com}_{let}$ are the same as those for **IMP** (with "$\rightarrow_{let}$" replacing "$\rightarrow$") with the addition of two rules for **letrec**. Let $\mathbf{lr}_i$ abbreviate the command $\mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, \mathbf{p}_i$. The first rule is:

$$\frac{\langle p'[\mathbf{lr}_1/\mathbf{p}_1] \ldots [\mathbf{lr}_n/\mathbf{p}_n], \sigma\rangle \rightarrow_{let} \sigma'}{\langle \mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, p', \sigma\rangle \rightarrow_{let} \sigma'} \qquad (\text{letrec-bind} \rightarrow_{let})$$

providing $p' \not\equiv \mathbf{p}_i$ for $1 \leq i \leq n$.

Here $p'[\mathbf{lr}_i/\mathbf{p}_i]$ denotes syntactic *substitution* of $\mathbf{lr}_i$ into p' for all occurrences of $\mathbf{p}_i$ which are not within a **letrec** construct of p', *viz.*, $\mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, \mathbf{p}_i$ replaces the "free" occurrences of $\mathbf{p}_i$ in p'.

The other rule is:

$$\frac{\left\langle p_i[\vec{\mathbf{lr}}_n/\vec{\mathbf{p}}_n], \sigma\right\rangle \rightarrow_{let} \sigma'}{\langle \mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, \mathbf{p}_i, \sigma\rangle \rightarrow_{let} \sigma'} \qquad \text{(letrec-unwind } \rightarrow_{let})$$

For example, let ***add*** be the open command

if $X \leq 0$
  then $Z := Y$
  else $X := X - 1;\ \mathbf{p}_1;\ X := X + 1;\ Z := Z + 1$

Problem 1.

1(a). Prove that $\mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1 \sim \mathbf{if}\, X \leq 0\, \mathbf{then}\, Z := Y\, \mathbf{else}\, Z := X + Y$ where $\sim$ is command equivalence (Winskel §2.4.1), and X, Y, and Z are distinct locations. *Hint*: By induction on $n = \sigma(X)$.

1(b). What can go wrong if X, Y, Z happen not be distinct?

Similarly, we could let *mult* be

if $X' \leq 0$
  then $Z' := 0$
  else $X' := X' - 1;\ \mathbf{p}_2;\ X' := X' + 1;\ X := Z';\ \mathbf{p}_1;\ Z' := Z$

and show that

$$(\mathbf{letrec}\, add, mult\, \mathbf{in}\, \mathbf{p}_2);\ X := 0;\ Z := 0 \sim$$
$$(\mathbf{if}\, X' \leq 0\, \mathbf{then}\, Z' := 0\, \mathbf{else}\, Z' := X' \times Y);\ X := 0;\ Z := 0,$$

again assuming X', Z', X, Y, Z are all distinct.

A more mnemonic syntax for this example might have been

$$\mathbf{letrec}\ \mathbf{p}_1\ \mathbf{be}\ add,\ \mathbf{p}_2\ \mathbf{be}\ mult\ \mathbf{in}\ \mathbf{p}_2,$$

but this style might encourage writing an illegal "nested open declaration" like

$$\mathbf{letrec}\ \mathbf{p}_1\ \mathbf{be}\ add\ \mathbf{in}\,(\mathbf{letrec}\ \mathbf{p}_2\ \mathbf{be}\ mult\ \mathbf{in}\ \mathbf{p}_2).$$

This would be disallowed according to the $\mathbf{IMP}_{let}$ well-formedness condition: *mult* has a free occurrence of $\mathbf{p}_1$, so the inner **letrec** phrase above would not be a (closed) command. Many languages would allow such nested declarations; others, such as the programming language C, don't. Our design decision for $\mathbf{IMP}_{let}$ to allow nesting of closed commands but not open ones, is unusual. It was made for pedagogical reasons: it gives a rich language but allows for simpler definitions of syntactic substitution and fixed points.

Problem 2.

2(a). Describe how to translate any $\mathbf{IMP}_{let}$ command into an equivalent $\mathbf{IMP}_{let}$ command which does not contain any **while** constructs. *Hint*: Show how to eliminate an "innermost" **while**.

2(b). Illustrate your method by translating

$$
\begin{array}{l}
\mathbf{letrec}\ \mathbf{while}\, b_1\ \mathbf{do}\ (X := X + 1; \mathbf{p}_2),\\
\qquad Y := Y + 1;\ \mathbf{p}_1\\
\mathbf{in}\\
\qquad \mathbf{while}\ b_2\ \mathbf{do}\\
\qquad\quad \mathbf{p}_1;\\
\qquad\quad \mathbf{letrec}\ Y := Y - 1\\
\qquad\quad \mathbf{in} \qquad \mathbf{while}\, b_3\, \mathbf{do}\, \mathbf{p}_1
\end{array}
$$

If we were given a (closed) command c_i for each $\mathbf{p}_i$ in an open command p, then p could be taken to correspond to the (closed) command $p[\vec{c}_i/\vec{\mathbf{p}}_i]$. For example, if

$$p \equiv \mathbf{if}\, b\, \mathbf{then}\, \mathbf{p}_2\, \mathbf{else}\, \mathbf{p}_3; c,$$

and if $\mathbf{p}_i$ corresponded to c_i for $i = 2, 3$, then p simply would correspond to $\mathbf{if}\, b\, \mathbf{then}\, c_2\, \mathbf{else}\, c_3; c$.

More abstractly, if we were given only the *meanings* (functions on states) of the procedure identifiers, then the open command would specify a new command meaning. For example, if the meaning of $\mathbf{p}_i$ was $\varphi_i : \Sigma \rightharpoonup \Sigma$ for $i = 2, 3$, then the p above would determine the state mapping φ where

$$\varphi(\sigma) = \begin{cases} \varphi_2(\sigma) & \text{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{true}, \\ \mathcal{C}[\![c]\!](\varphi_3(\sigma)) & \text{otherwise.} \end{cases}$$

In general, we can understand any open command p as denoting a function Γ_p from any n-vector of command meanings to a command meaning, providing $n \geq \max\{i \mid \mathbf{p}_i \text{ occurs in } p\}$. In other words, for the denotational semantics $\mathcal{O}[\![\cdot]\!]$ of **OCom** we have

$$\mathcal{O}[\![p]\!] = \Gamma_p : (\Sigma \rightharpoonup \Sigma)^n \to (\Sigma \rightharpoonup \Sigma),$$

where the key property of Γ_p is

$$\Gamma_p(\mathcal{C}[\![c_1]\!], \ldots, \mathcal{C}[\![c_n]\!]) = \mathcal{C}[\![p[\vec{c}_n/\vec{\mathbf{p}}_n]]\!]$$

for all commands $\vec{c}_n$.

Now a *well-formed* vector $\vec{p}_n$ of open commands, namely a vector such that $n \geq \max\{i \mid \mathbf{p}_i \text{ occurs in } \vec{p}_n\}$, can be understood as defining a function $\Gamma_{\vec{p}_n}$ from a vector command meanings $\vec{\varphi}_n$ to another vector of command meanings. Namely,

$$\Gamma_{\vec{p}_n} : (\Sigma \rightharpoonup \Sigma)^n \to (\Sigma \rightharpoonup \Sigma)^n,$$

$$\Gamma_{\vec{p}_n}(\vec{\varphi}_n) = (\Gamma_{p_1}(\vec{\varphi}_n), \ldots, \Gamma_{p_n}(\vec{\varphi}_n)).$$

Then we define $\mathcal{C}[\![\cdot]\!]$ for $\mathbf{Com}_{let}$ just as for **IMP** commands, with one more case for **letrec**:

$$\mathcal{C}[\![\mathbf{letrec}\, \vec{p}_n \,\mathbf{in}\, p']\!] = \Gamma_{p'}(\mathit{fix}(\Gamma_{\vec{p}_n})).$$

The least fixed point of $\Gamma_{\vec{p}_n}$ exists because it is continuous, as you will verify in the next problems.

Problem 3. Carefully define Γ_p. *Hint*: Structural induction on **OCom**.

Problem 4.

4(a). Show that Γ_p is continuous.

4(b). Conclude that for well-formed $\vec{p}_n$, the function $\Gamma_{\vec{p}_n}$ is continuous.

We close this problem set with the remark that the proof of the equivalence of the operational and denotational semantics for **IMP** in Winskel §5.3, and for $\mathbf{IMP}_v$ in Problem Set 4 carries over to $\mathbf{IMP}_{let}$ with one significant change. Because $\mathbf{IMP}_{let}$ commands contain open commands as subterms, a proof by structural induction that $\mathcal{C}[\![c]\!]\sigma = \sigma'$ implies $\langle c, \sigma\rangle \to_{let} \sigma'$ requires a more general structural induction on open commands. The "right" induction hypothesis for open commands is not obvious. Define a command c to be *OK* if $\mathcal{C}[\![c]\!]\sigma = \sigma'$ implies $\langle c, \sigma\rangle \to_{let} \sigma'$ for all σ, σ'. Define an *open* command p to be OK if $p[\vec{c}_n/\vec{\mathbf{p}}_n]$ is OK whenever c_i is OK for $1 \leq i \leq n$. With this hypothesis, it is not too hard to prove by structural induction that all open commands are OK. Finally, because (closed) commands are a special case of open commands, we conclude that all commands are OK.

We won't write this up more fully, but state that, as expected,

$$\langle c, \sigma\rangle \to_{let} \sigma' \text{ iff } \mathcal{C}[\![c]\!](\sigma) = \sigma'.$$

Problem Set 4 Solutions

Problem 1. Prove the claim made in Winskel, §5.2, p.54, l.-10 that $\Gamma = \widehat{R}$.

Solution: The text defined Γ as a (total) function from sets to sets. Specifically, if S is any set then the set Γ(S) is as follows:

$$\begin{aligned}\Gamma(\mathrm{S}) &= \{(\sigma,\sigma') \mid \exists\sigma''.\beta(\sigma) = \mathbf{true}\ \&\ (\sigma,\sigma') \in \gamma\ \&\ (\sigma'',\sigma') \in \mathrm{S}\} \\ &\quad \cup \{(\sigma,\sigma) \mid \beta(\sigma) = \mathbf{false}\},\end{aligned}$$

Note that $\beta(\sigma)$ was defined to be $\mathcal{B}[\![b]\!]\sigma$, and γ was defined to be $\mathcal{C}[\![c]\!]$. I only mention this to make things a little clearer, the proof that $\Gamma = \hat{R}$ does not depend on the actual definitions of β and γ *in any way*.

Also notice that in the definition of Γ, the binding of φ to $\mathcal{C}[\![w]\!]$ was released. Γ was defined as a function which given an arbitrary set φ returned the new set $\Gamma(\varphi)$. To make the definition clearer, I have renamed this bound varible φ to S.

The set of rule instances R was defined to be:

$$\begin{aligned}R &= \left\{\left(\{(\sigma'',\sigma')\}/(\sigma,\sigma')\right) \mid \beta(\sigma) = \mathbf{true}\ \&\ (\sigma,\sigma'') \in \gamma\right\} \\ &\quad \cup \{((\sigma,\sigma)) \mid \beta(\sigma) = \mathbf{false}\}.\end{aligned}$$

We can rewrite R in a more formaliar style, by stating two rule schema.

$$\frac{(\sigma'',\sigma')}{\sigma,\sigma'} \qquad (\text{true } \Gamma)$$

where $\beta(\sigma) = \mathbf{true}$ and $(\sigma,\sigma'') \in \gamma$.

$$(\sigma,\sigma) \qquad (\text{false } \Gamma)$$

where $\beta(\sigma) = \mathbf{false}$.

Now that we understand the definitions of Γ and R, showing that $\Gamma = \hat{R}$ (*i.e.* that for all sets S, $\Gamma(\mathrm{S}) = \hat{R}(S)$) is a trivial chugging through of the defition of $\hat{R}$. Remember the definition of $\hat{R}(\mathrm{S})$ from R:

$$\hat{R}(\mathrm{S}) = \{y \mid \exists X \subseteq \mathrm{S}.\ (X/y) \in R\}$$

To show that $\Gamma(S) = \hat{R}(S)$, we first argue that all elements of $\Gamma(S)$ must be elements of $\hat{R}(S)$. Then we argue the converse. We also note that all elements of either side must be of the form (σ, σ') (*i.e.* A pair of (possibly) distinct states).

Suppose $(\sigma, \sigma') \in \Gamma(S)$. Then either:

$$(\sigma, \sigma') \in \{(\sigma, \sigma') | \exists \sigma''. \beta(\sigma) = \mathbf{true} \ \& \ (\sigma, \sigma') \in \gamma \ \& \ (\sigma'', \sigma') \in S\}$$

or

$$(\sigma, \sigma') \in \{(\sigma, \sigma) | \beta(\sigma) = \mathbf{false}\}.$$

Suppose it is the first case. Then there exists a σ'' such that $(\sigma'', \sigma') \in S$, $\beta(\sigma) = \mathbf{true}$, and $(\sigma, \sigma'') \in \gamma$. And so by the rule (true Γ), (σ, σ') can be obtained by one application of a rule in R to a set of elements of S (namely $\{(\sigma'', \sigma')\}$), and so by the definition of $\hat{R}$, $(\sigma, \sigma') \in \hat{R}(S)$.

In the second case then we know that $\sigma' \equiv \sigma$, and $\beta(\sigma) = \mathbf{false}$. But then by rule (false Γ), (σ, σ') can be obtained by one application of a rule in R to a set of elements of S (namely $\emptyset$), and so by definiton of $\hat{R}$, $(\sigma, \sigma') \in \hat{R}(S)$.

We have now completed a proof that for an arbitrary set S, $\Gamma(S) \subseteq \hat{R}(S)$.

We now show that $\hat{R}(S) \subseteq \Gamma(S)$, thereby completing the proof.

Suppose that $(\sigma, \sigma') \in \hat{R}(S)$, then it must have gotten there by an instance of either (true Γ) or (false Γ). Suppose it got there by an instance of (true Γ). Then, by the form of this rule, and the definition of $\hat{R}$, there must be a σ'', such that $(\sigma'', \sigma') \in S$, $\beta(\sigma) = \mathbf{true}$, and $(\sigma, \sigma'') \in \gamma$. But then,

$$(\sigma, \sigma') \in \{(\sigma, \sigma') | \exists \sigma''. \beta(\sigma) = \mathbf{true} \ \& \ (\sigma, \sigma') \in \gamma \ \& \ (\sigma'', \sigma') \in S\}$$

and so by the definition of $\Gamma(S)$, $(\sigma, \sigma') \in \Gamma(S)$.

Suppose $(\sigma, \sigma') \in \hat{R}(S)$ by the rule (false Γ). Then $\sigma' \equiv \sigma$, and $\beta(\sigma) = \mathbf{true}$, and so,

$$(\sigma, \sigma') \in \{(\sigma, \sigma) | \beta(\sigma) = \mathbf{false}\}.$$

Then by the definition of $(\Gamma(S)$, $(\sigma, \sigma') \in \Gamma(S)$, and we are done.

Problem 2.

2(a). Exhibit $a_0, a_1 \in \mathbf{Aexp}_r, n_0 \neq n_1 \in \mathbf{Num}, \sigma, \sigma' \in \Sigma$ such that

$$\langle a_0 + a_1, \sigma \rangle \rightarrow_r \langle n_0, \sigma' \rangle$$

and

$$\langle a_1 + a_0, \sigma \rangle \rightarrow_r \langle n_1, \sigma' \rangle.$$

(Addition is not commutative in $\mathbf{IMP}_r$ in that $n_0 \neq n_1$).

Solution: Here is a good example: $a_0 \equiv X$, $a_1 \equiv ((X := X+1)\,\mathbf{resultis}\,5)$. In a state σ, such that $\sigma(X) = 100$, then $\langle a_0 + a_1, \sigma\rangle \rightarrow_r \langle 105, \sigma[101/X]\rangle$, whereas $\langle a_1 + a_2, \sigma\rangle \rightarrow_r \langle 106, \sigma[101/X]\rangle$.

2(b). Outline a proof that for all $a_0, a_1 \in \mathbf{Aexp}_v, n \in \mathbf{Num}, \sigma \in \Sigma$ that

$$\langle a_0 + a_1, \sigma\rangle \rightarrow_v n \text{ iff } \langle a_1 + a_0, \sigma\rangle \rightarrow_v n.$$

(So addition is commutative in $\mathbf{IMP}_v$.)

Solution: This problem was a lot easier than everyone made it out to be. There is no induction involved it all!! It is just a matter of cutting and pasting derivations similar to the proof of:

$$\mathbf{while}\, b\, \mathbf{do}\, c \sim \mathbf{if}\, b\, \mathbf{then}(c; \mathbf{while}\, b\, \mathbf{do}\, c)\, \mathbf{else}\, \mathbf{skip}$$

We show, for arbitrary $a_0, a_1 \in \mathbf{Aexp}$, $\sigma \in \Sigma$ and $n \in \mathbf{Num}$, that

$$\langle a_0 + a_1, \sigma\rangle \rightarrow_v n \text{ implies } \langle a_1 + a_1, \sigma\rangle \rightarrow_v n$$

Note: because a_0 and a_1 were arbitrary, showing this implication in fact shows the "iff".

So, suppose $\langle a_0 + a_1, \sigma\rangle \rightarrow_v n$. Then there must be a derivation of this. Looking at the rules for $\rightarrow_v$, we see that the derivation must take the following form:

$$\dfrac{\left.\dfrac{\vdots}{\langle a_0, \sigma\rangle \rightarrow_v n_0}\right\} D_0 \quad D_1 \left\{\dfrac{\vdots}{\langle a_1, \sigma\rangle \rightarrow_v n_1}\right.}{\langle a_0 + a_1, \sigma\rangle \rightarrow_v n}$$

where the sum of n_0 and n_1 is n. So, D_0 is a derivation of $\langle a_0, \sigma\rangle \rightarrow_v n_0$, and D_1 is a derivation of $\langle a_1, \sigma\rangle \rightarrow_v n_1$. But by reversing the roles of a_0 and a_1, we can use the rule (plus $\rightarrow_v$) to obtain the derivation:

$$\dfrac{\left.\dfrac{\vdots}{\langle a_1, \sigma\rangle \rightarrow_v n_1}\right\} D_1 \quad D_0 \left\{\dfrac{\vdots}{\langle a_0, \sigma\rangle \rightarrow_v n_0}\right.}{\langle a_1 + a_0, \sigma\rangle \rightarrow_v n}$$

Which is a legal derivation because we already had the legal derivations D_1 and D_0, and because the last step is a legal application of (plus $\rightarrow_v$). One part of verifying the legality of this rule application is to check that the sum of n_1 and n_0 is n. This follows trivially, from the fact that we already have that the sum of n_0 and n_1 is n, and that addition is commutative. It is in this very last step that chugging this proof through for "$-$" would fail, as it should.

Problem 3. Let $\hookrightarrow_v$ be defined by adding the (par-if) rule to the rules for $\rightarrow_v$ as done on Quiz 1 for **IMP**.

$$\frac{\langle c_0, \sigma\rangle \hookrightarrow_v \sigma', \quad \langle c_1, \sigma\rangle \hookrightarrow_v \sigma'}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, \sigma\rangle \hookrightarrow_v \sigma'} \qquad \text{(par-if } \hookrightarrow_v\text{)}$$

3(a). Briefly sketch how to prove that $\langle c, \sigma\rangle \rightarrow_v \sigma'$ implies $\langle c, \sigma\rangle \hookrightarrow_v \sigma'$.

Solution on Attached Page.

3(b). Give a simple **IMP**$_v$ command configuration which is a counterexample to the claim that $\hookrightarrow_v$ implies $\rightarrow_v$. Briefly explain where the proof on Quiz 1 of the corresponding implication for **IMP** breaks down for **IMP**$_v$.

Solution on Attached Page.

Problem 4.

4(a). Give the definition of a denotational semantics for **IMP**$_v$ by structural induction. (Your definition should satisfy the result of problem 4(b).)

Solution: As mentioned in class, for **IMP**$_v$ it is not possible to seperate structural induction on **Aexp**'s from structural induction on **Com**'s since **Aexp**'s might now contain **Com**'s , and since **Com**'s may contain **Bexp**'s and **Bexp**'s can contain **Aexp**'s. We typically will need to define or prove something about **IMP**$_v$ code considering all **Aexp**'s, **Bexp**'s and **Com**'s at the same time.

For the definition of $\mathcal{A}[\![a]\!]$, $\mathcal{B}[\![b]\!]$, and $\mathcal{C}[\![c]\!]$, all of the cases are defined to be as they were for **IMP**, except we obviously must add the case of $a \equiv c_0\, \mathbf{valis}\, a_0$. Which is

$$\mathcal{A}[\![c_0\, \mathbf{valis}\, a_0]\!] = \{(\sigma, n) | \exists \sigma'.(\sigma, \sigma') \in \mathcal{C}[\![c]\!]\ \&\ \mathcal{A}[\![a_0]\!]\sigma' = n\}$$

In addition since we are doing this more complex induction, we observe that the only base cases in a structural induction on $\alpha \in$ **IMP**$_v$ are: $\alpha \equiv n$, $\alpha \equiv t$, and $\alpha \equiv \mathbf{skip}$.

All other forms of **IMP**$_v$ code, are no longer base cases.

4(b). The proof of the equivalence of the operational and denotational semantics for **IMP** in Winskel §5.3 carries over to **IMP**$_v$ with only minor changes. Briefly, but clearly, indicate the changes needed in the proof in §5.3 to prove that for all $c \in$ **Com**$_v$,

$$\langle c, \sigma\rangle \rightarrow_v \sigma' \text{ iff } \mathcal{C}[\![c]\!](\sigma) = \sigma'.$$

Joe Nantz
6.044J
10/18/91

10/10

Problem Set #4

Problem 3

(3a) This proof is very similar to the ~~proof~~ on Quiz 1. The implication,

$$\langle c, \sigma \rangle \to_v \sigma' \text{ implies } \langle c, \sigma \rangle \hookrightarrow_v \sigma' \quad \text{can}$$

be proved by induction on the definition of $\to_v$. There is one case for each of the inference rules of $\to_v$ on Com-configurations.

Base case rules: (skip $\to_v$), (assign $\to_v$)

Inductive cases rules: (seq $\to_v$), (if-true $\to_v$), (if-false $\to_v$) (while-true $\to_v$), (while-false $\to_v$).

If $\langle c, \sigma \rangle \to_v \sigma'$ follows from some ($\to_v$)-rule, R, then the antecedents if any of R which involve $\to_v$, each implies by induction, the corresponding antecedent with "$\to_v$" replaced by "$\hookrightarrow_v$", so $\langle c, \sigma \rangle \hookrightarrow_v \sigma'$ follows trivially by the $\hookrightarrow_v$-version of R.

Note: The rule

$$\frac{\langle c_0, \sigma \rangle \hookrightarrow_v \sigma', \quad \langle c_1, \sigma \rangle \hookrightarrow_v \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma \rangle \hookrightarrow_v \sigma'}$$

never comes into play because an equivalent rule does not exist in $\to_v$. In some sense, "$\to_v$" could be a subset of "$\hookrightarrow_v$". As a result, $\to_v$ implies $\hookrightarrow_v$. Excellent!!

(3b) $\langle \text{if } b \text{ then } c' \text{ else } c', \sigma \rangle \hookrightarrow_v \sigma'$ where

$b \equiv (((\text{while true do skip}) \text{ valis } 1) = 1)$ is a counterexample to the claim that $\hookrightarrow_v$ implies $\rightarrow_v$. The proof on Quiz I breaks down in the nontrivial case for IMP_v. The expression above would evaluate ~~to~~ σ' under the (par-if $\hookrightarrow_v$) rule because b is never evaluated. However, the same expression would not evaluate given any $\rightarrow_v$ rule. The evaluation of (while true do skip) in the valis command would loop for infinity. Thus, here is an example where $\hookrightarrow_v$ doesn't imply $\rightarrow_v$. great!!

Solution: Here the story is somewhat complicated. We combine half of Lemmas 12 and 13, with Lemma 15, and prove by structural induction on $\alpha \in \mathbf{IMP}_v$, that $P(\alpha)$ holds, where $P(\alpha)$ is defined to be:

$$\begin{array}{llll} \forall\sigma, n.\mathcal{A}[\![a]\!]\sigma = n & \text{implies} & \langle a, \sigma\rangle \rightarrow_v n & \text{if } \alpha \equiv a \in \mathbf{Aexp} \\ \forall\sigma, t.\mathcal{B}[\![b]\!]\sigma = t & \text{implies} & \langle c, \sigma\rangle \rightarrow_v t & \text{if } \alpha \equiv b \in \mathbf{Bexp} \\ \forall\sigma, \sigma'.(\sigma, \sigma') \in \mathcal{C}[\![c]\!] & \text{implies} & \langle c, \sigma\rangle \rightarrow_v \sigma' & \text{if } \alpha \equiv c \in \mathbf{Com} \end{array}$$

Then all of our old base cases become inductive cases, except for $\alpha \equiv n, t, \mathbf{skip}$ as in part (a).

Showing $P(\alpha)$ holds for all $\alpha \in \mathbf{IMP}_v$ is at the top level a case analysis on whether $\alpha \in \mathbf{Aexp}, \mathbf{Bexp}$ or $\mathbf{Com}$. All of the work on the next level is the same as that for **IMP**, except, of course we also need to do the case of $\alpha \equiv c_0 \,\mathbf{valis}\, a_0$.

For this case we fix an arbitrary σ and n. We must show:

$$\mathcal{A}[\![c_0 \,\mathbf{valis}\, a_0]\!]\sigma = n \text{ implies } \langle c_0 \,\mathbf{valis}\, a_0, \sigma\rangle \rightarrow_v n$$

So, suppose $\mathcal{A}[\![c_0 \,\mathbf{valis}\, a_0]\!]\sigma = n$. Then by the definition of $\mathcal{A}[\![\cdot]\!]$, there exists a σ' such that $(\sigma, \sigma') \in \mathcal{C}[\![c_0]\!]$, and $\mathcal{A}[\![a_0]\!]\sigma' = n$. By induction we have $P(c_0)$ and $P(a_0)$, so $\langle c_0, \sigma\rangle \rightarrow_v \sigma'$, and $\langle a_0, \sigma'\rangle \rightarrow_v n$. Finally, by rule (valis $\rightarrow_v$) $\langle c_0 \,\mathbf{valis}\, a_0, \sigma\rangle \rightarrow_v n$.

A separate induction (this time a rule-induction) captures the other halves of Lemmas 12, 13, and all of Lemma 14. Here define the property $P(\alpha, \sigma, \gamma)$ by:

$$\begin{array}{llll} \langle a, \sigma\rangle \rightarrow_v n & \text{implies} & \mathcal{A}[\![a]\!]\sigma = n & \text{if } \alpha \equiv a \in \mathbf{Aexp}, \gamma \equiv n \in \mathbf{Num} \\ \langle b, \sigma\rangle \rightarrow_v t & \text{implies} & \mathcal{B}[\![b]\!]\sigma = t & \text{if } \alpha \equiv b \in \mathbf{Bexp}, \gamma \equiv t \in \{\mathbf{true}, \mathbf{false}\} \\ \langle c, \sigma\rangle \rightarrow_v \sigma' & \text{implies} & (\sigma, \sigma') \in \mathcal{C}[\![c]\!] & \text{if } \alpha \equiv c \in \mathbf{Com}, \gamma \equiv \sigma' \in \Sigma \end{array}$$

Again, we have our top level case analysis depending whether α is an **Aexp**, **Bexp**, or **Com**. There is the same shifting of base cases to inductive cases, leaving only the three base cases as before. Finally, what was a structural induction for **Aexp**'s and **Bexp**'s has become a rule induction. The only appreciable difference within each case is the precise manner in which the induction hypothesis is invoked. But it works well enough.

In addition we need to add our case of $\alpha \equiv c_0 \,\mathbf{valis}\, a_0$, $\gamma = n$. Suppose $\langle c_0 \,\mathbf{valis}\, a_0, \sigma\rangle \rightarrow_v n$, this can only happen by the rule (valis $\rightarrow_v$), and so we know that there exists a σ' such that $\langle c_0, \sigma\rangle \rightarrow_v \sigma'$ and $\langle a_0, \sigma'\rangle \rightarrow n$. So, by induction $P(c_0, \sigma, \sigma')$ and $P(a_0, \sigma, n)$, thus $(\sigma, \sigma') \in \mathcal{C}[\![c_0]\!]$, and $\mathcal{A}[\![a_0]\!]\sigma' = n$. Finally, by the definition of $\mathcal{A}[\![\cdot]\!]$, we have that $\mathcal{A}[\![c_0 \,\mathbf{valis}\, a_0]\!]\sigma = n$.

Addendum to Problem Set 5

All of the definitions in Revised Problem Set 5 (handout 20) still hold.

Due: 25 October 1991

All definitions remain as in Handout 20

Problems 1 and 2 remain unchanged.

Revised Problem 3 Γ_p can be defined by an induction on the structure of **OCom**. Setting up this induction has some subtle points. In particular consider the case of $p \equiv \mathbf{while}\, b\, \mathbf{do}\, p_0$. For this case have:

$$\Gamma_{\mathbf{while}\, b\, \mathbf{do}\, p_0}(\vec{\varphi}_n) = \mathit{fix}(\Gamma'),$$

where Γ' is defined by:

$$\Gamma'(\psi)\sigma = \begin{cases} \sigma & \text{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{false}, \\ \psi(\Gamma_{p_0}(\vec{\varphi}_n)\sigma) & \text{otherwise.} \end{cases}$$

The goal of this problem is to show that $\Gamma_{\mathbf{while}\, b\, \mathbf{do}\, p_0}$ is well-defined (assumming that Γ_{p_0}) is well defined. To do so we show that $\mathit{fix}(\Gamma')$ is well-defined.

3(a). Prove that $\Gamma' : (\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$ is continuous. *i.e.* show that if $\psi_0, \psi_1, \ldots$ is an ascending chain in $\Sigma \rightharpoonup \Sigma$, then

$$\Gamma'\left(\bigsqcup_{n\geq 0} \psi_n\right) = \bigsqcup_{n\geq 0} (\Gamma'(\psi_n))$$

Recall that we order $\Sigma \rightharpoonup \Sigma$ under $\subseteq$.

3(b). Show that Γ' has a least fixpoint, and thus conclude that $\mathit{fix}(\Gamma')$ is well defined.

Revised Problem 3-OPT. (OPTIONAL) Do the original Problem 3 from Problem Set 5.

Revised Problem 4-OPT. (OPTIONAL) Do the original Problem 4 from Problem Set 5.

Some Exercises on CPO's

Included are 4 exercises taken from Chapter 6 section 3 of the class handouts from 6.821 this term. The solutions to exercises 6.7 and 6.9 are taken from 6.821 handout #14 from 5 October 1990, which was entitled "Problem Set 3 Solution."

These exercises use "domain" in two different ways. A function is a maf from its domain to its range. And a cpo with a bottom may also be a domain. It should always be clear from the context which is intended.

These should give you an idea of the level of understanding which we expect for the quiz.

Exercise 1. Try exercises 6.6 to 6.9 taken from 6.821, they are on attached pages.

Exercise 2. Try and prove Theorem 16 on page 64 of Winskel, without looking at the proof there.

2

▷ **Exercise 6.6** Consider the lifted flat domain $D = \{a, b, c\}_\perp$. How many monotonic functions are there from D to D? ♡

▷ **Exercise 6.7** Suppose the domains E and F are defined as follows:

$$\begin{aligned} E &= \{a, b\} \quad \text{where } a \sqsubseteq b \\ F &= \{c, d\} \quad \text{where } c \text{ and } d \text{ are incomparable} \end{aligned}$$

Consider the domain $E_\perp \rightarrow F_\perp$. The elements of $E_\perp \rightarrow F_\perp$ are themselves related by a partial order. Draw the partial order whose elements are the members of $E_\perp \rightarrow F_\perp$, where you represent a function in $E_\perp \rightarrow F_\perp$ by its graph. (*Hint:* for an example of partial orders on functions, see page 119 of Schmidt.) ♡

▷ **Exercise 6.8** In the function descriptions given below, we specify on the left-hand side of the = the name and the of each function. The signature describes the domain and range of the function. For example, the familiar > function on the natural numbers, which tests whether its first argument is strictly greater than its second, has the signature:

$$> : (\mathit{Integer} \times \mathit{Integer}) \rightarrow \mathit{Boolean}$$

For each function specified below, say whether or not the function is monotonic. Briefly explain your answer. (Recall that *Whole-Number* is the flat domain of whole numbers.)

a. $f_1 : \mathit{Whole\text{-}Number}_\perp \rightarrow \mathit{Whole\text{-}Number}_\perp = \lambda n \,.\, 3$

b. $f_2 : \mathit{Whole\text{-}Number}_\perp \rightarrow \mathit{Whole\text{-}Number}_\perp = \lambda n \,.\, 3$

c. $f_3 : \mathit{Whole\text{-}Number}_\perp \rightarrow \mathit{Whole\text{-}Number}_\perp = \lambda n \,.\, (n = \perp) \rightarrow 3 \,[\!]\, (n+1)$

d. $f_4 : (\mathit{Whole\text{-}Number}_\perp \rightarrow \mathit{Whole\text{-}Number}_\perp) \rightarrow (\mathit{Whole\text{-}Number}_\perp \rightarrow \mathit{Whole\text{-}Number}_\perp) =$
$\lambda g \,.\, \lambda n \,.\, (n = \perp) \rightarrow 3 \,[\!]\, g(n)$

$\lambda n. \{\perp$ if $n=\perp$; 3 otherwise$\}$

$\lambda n. \{3$ if $n=\perp$; $n+1$ otherwise.$\}$

$\lambda g.\lambda n. \{3$ if $n=\perp$; $g(n)$ otherwise$\}$

Integer = Num

Whole-Number = {0, 1, 2, ...}

Boolean = {true, false}

▷ **Exercise 6.9**

a. Suppose that the result of the application (**fix** x) has the signature $A \rightarrow B$. What is the signature of x?

b. Assuming that the functional domain $A \rightarrow B$ is a pointed cpo, what conditions must be placed on x so that (**fix** x) exists?

c. Consider the function *FACT* : $\mathit{Whole\text{-}Number}_\perp \rightarrow \mathit{Whole\text{-}Number}_\perp$ whose graph is given by

$$(\text{graph } \mathit{FACT}) = \{\langle\perp, \perp\rangle\} \cup \{\langle n, n!\rangle \mid n \in \mathit{Whole\text{-}Number}\}$$

What is (*fix FACT*)?

Answers to Exercise 1

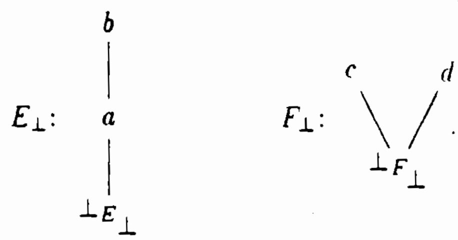
6.6 If we map ⊥ to ⊥ then we allow all ~~8~~ $4^3 = 12$ functions from {a, b, c} to $\{a, b, c\}_\perp$.

There is only 1 function which can map ⊥ to a (the constant a function)

Similarly there are 2 other functions

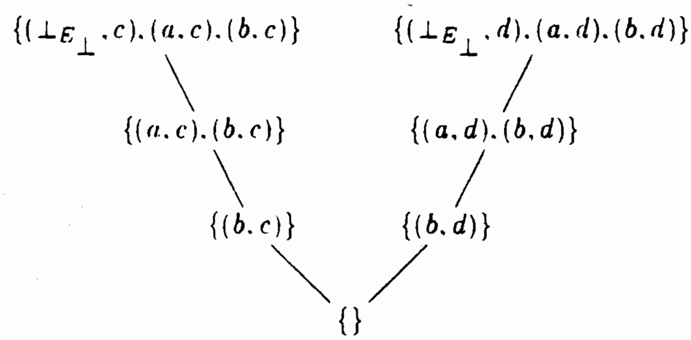
For a total of 15.

6.7 The domains $E_\perp$ and $F_\perp$ look as follows:



Recall that the domain $E_\perp - F_\perp$ contains all *continuous* functions from $E_\perp$ to $F_\perp$. For function domains $X - Y$ where X is finite, continuous and monotonic mean exactly the same thing (convince yourself of this). Out of the $3^3 = 27$ possible functions between $E_\perp$ and $F_\perp$, only 7 are monotonic. These 7 are pictured below as a partial ordering on functions. As in Schmidt's notation, we have elided pairs whose second element is ⊥: the key advantage of this notation is that the partial order on functions is the same as the partial order on their graphs induced by the subset relation.

$E_\perp - F_\perp$:

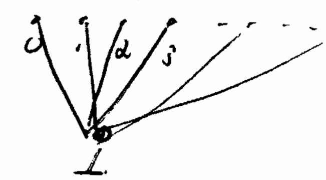


6.8

a. Monotonic. $f_1(n_1) \sqsubseteq f_1(n_2)$ for all n_1, n_2 since
$f_1(n_1) = f_1(n_2) = 3$.

b. monotonic. why: Suppose $n_1 \sqsubseteq n_2$ then either $n_1 = n_2$, in which case we are done;
or $n_1 = \bot$ in which case $f(n_1) = \bot$ which clearly is $\sqsubseteq f(n_2)$

c. Not monotonic. $\bot \sqsubseteq 4$
but $f(\bot) \not\sqsubseteq f(4)$
$3 \not\sqsubseteq 5$ (remember Whole-Number$_\bot$ looks like



So $0 \not\sqsubseteq 1$!

d. Monotonic! (suprise!) f_4 is a monotonic function from
(Whole-number$_\bot$ → Whole-number$_\bot$) ← the set of all functions from $W_\bot$ to $W_\bot$
to (Whole-number$_\bot$ → Whole-number$_\bot$) ← the set of all functions from $W_\bot$ to $W_\bot$
if $g_1 \sqsubseteq g_2$ then $f(g_1) \sqsubseteq f(g_2)$.
but Notice $f(g_1)$ might not be monotonic!
eg if $g_1 = \lambda n.\, n+1$ then $f_3 = f_4(g_1)$.

6.9 (5)

- Suppose $fix\ x : (A \rightarrow B)$. Fix computes the least fixed point of x, so by definition of a fixed point,

$$fix\ x = x\,(fix\,x)$$

Suppose $y = fix\,x$. Since x can be applied to y, then x must be a function whose inputs come from the domain which is the signature of y. That is $x : (A \rightarrow B) \rightarrow$ something. But we know that $x\ y = y$, so x applied to something results in elements of $A \rightarrow B$. Therefore,

$$x : (A \rightarrow B) \rightarrow (A \rightarrow B)$$

- ~~(You did not have to answer this part.)~~ If $A \rightarrow B$ is a pointed CPO, then $fix\ x$ exists if x is continuous, i.e. if x is monotonic and preserves least upper bounds of chains.

- Careful! The answer is *not* the factorial function. The signature of *fix* over a domain D is

$$fix_D : (D \rightarrow D) \rightarrow D$$

For defining recursive functions, we take the fixed point over a function domain, i.e. $D = A \rightarrow B$. But *fix* works over any D that is a continuous pointed CPO. The graph given in this problem is the graph of the factorial function, not a function that generates factorial approximations (like FACT-EQN in the notes). This function has 3 fixed points: $\bot$, 1, and 2. $\bot \sqsubseteq 1$ and $\bot \sqsubseteq 2$ (note 1 and 2 are not comparable), so the least fixed point is $\bot$.

6.3 Domain Theory

This section is under development. For more information about domain theory, including complete partial orders, least fixed points, and recursive domains, please consult the following:

- Schmidt, David. *Denotational Semantics: A Methodology for Language Development.* Allyn and Bacon, 1986. Of particular interest are chapters 2 & 3 (domains), chapter 6 (recursive functions), and chapter 11 (recursive domains).
- Stoy, Joseph. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* M.I.T. Press., 1977. See especially chapters 2 & 3 (introduction to denotational semantics), chapter 6 (lattices and domains), and chapter 7 (recursive domains).
- Tennent, R. D. "The Denotational Semantics of Programming Languages." *Communications of the ACM*, Volume 9, Number 8, August 1976, pp. 437-452. A tutorial paper explaining the fundamentals of denotational semantics.

▷ **Exercise 6.6** Consider the lifted flat domain $D = \{a, b, c\}_\bot$. How many monotonic functions are there from D to D? ♡

▷ **Exercise 6.7** Suppose the domains E and F are defined as follows:

$$\begin{array}{rcll} E & = & \{a, b\} & \text{where } a \sqsubseteq b \\ F & = & \{c, d\} & \text{where } c \text{ and } d \text{ are incomparable} \end{array}$$

Consider the domain $E_\bot \rightarrow F_\bot$. The elements of $E_\bot \rightarrow F_\bot$ are themselves related by a partial order. Draw the partial order whose elements are the members of $E_\bot \rightarrow F_\bot$, where you represent a function in $E_\bot \rightarrow F_\bot$ by its graph. (*Hint:* for an example of partial orders on functions, see page 119 of Schmidt.) ♡

▷ **Exercise 6.8** In the function descriptions given below, we specify on the left-hand side of the = the name and the of each function. The signature describes the domain and range of the function. For example, the familiar > function on the natural numbers, which tests whether its first argument is strictly greater than its second, has the signature:

$$> : (\mathit{Integer} \times \mathit{Integer}) \rightarrow \mathit{Boolean}$$

For each function specified below, say whether or not the function is monotonic. Briefly explain your answer. (Recall that *Whole-Number* is the flat domain of whole numbers.)

a. $f_1 : \textit{Whole-Number}_\bot \rightarrow \textit{Whole-Number}_\bot = \lambda n \,.\, 3$

b. $f_2 : \textit{Whole-Number}_\bot \rightarrow \textit{Whole-Number}_\bot = \underline{\lambda} n \,.\, 3$

c. $f_3 : \textit{Whole-Number}_\bot \rightarrow \textit{Whole-Number}_\bot = \lambda n \,.\, (n = \bot) \rightarrow 3 \,[\!]\, (n + 1)$

d. $f_4 : (\textit{Whole-Number}_\bot \rightarrow \textit{Whole-Number}_\bot) \rightarrow (\textit{Whole-Number}_\bot \rightarrow \textit{Whole-Number}_\bot) = \lambda g \,.\, \lambda n \,.\, (n = \bot) \rightarrow 3 \,[\!]\, g(n)$

♡

▷ **Exercise 6.9**

a. Suppose that the result of the application (**fix** x) has the signature $A \rightarrow B$. What is the signature of x?

b. Assuming that the functional domain $A \rightarrow B$ is a pointed cpo, what conditions must be placed on x so that (**fix** x) exists?

c. Consider the function *FACT* : *Whole-Number*$_\perp$ $\rightarrow$ *Whole-Number*$_\perp$ whose graph is given by

$$(graph\ FACT) = \{\langle \perp,\ \perp \rangle\} \cup \{\langle n,\ n! \rangle \mid n \in \mathit{Whole\text{-}Number}\}$$

What is (*fix FACT*)?

♡

▷ **Exercise 6.10** Suppose that $n \in$ *Integer* and that $f \in$ *Integer* $\rightarrow$ *Integer*$_\perp$. Also assume that +, -, *, /, =, <, *square*, *even?* have their usual meanings on the integers.

For each of the functions below:

1. Characterize all of its fixed points.
2. Indicate which of its fixed points is the least fixed point.

Example:

```
λf. λn. if (= n 0)
        then 1
        else if (< n 0)
             then (f (+ n 1))
             else (f n)
             endif
        endif
```

1. For any choice of $c \in$ *Integer*$_\perp$, the function f_c whose graph is

$$\{\langle n,\ 1 \rangle \mid n \leq 0\} \cup \{\langle n,\ c \rangle \mid n > 0\}$$

is a fixed point of the above function.

2. The least fixed point is $f_\perp$.

a. $\lambda f.\ \lambda n.$ (*square* (+ n *1*))

b. $\lambda f.\ \lambda n.$ (*square* n)

Problem Set 5 Solutions

Problem 1.

1(a). Prove that **letrec** *add* **in** $\mathbf{p}_1 \sim$ **if** $X \leq 0$ **then** $Z := Y$ **else** $Z := X + Y$ where $\sim$ is command equivalence (Winskel §2.4.1), and X, Y, and Z are distinct locations. *Hint*: By induction on $n = \sigma(X)$.

Solution:

We first recall the appropriate definition of $c_0 \sim c_1$ for $\mathbf{IMP}_{let}$. It says, for all states σ, σ',

$$\langle c_0, \sigma\rangle \rightarrow_{let} \sigma' \text{ iff } \langle c_1, \sigma\rangle \rightarrow_{let} \sigma'$$

We will actually prove this by establishing two Lemmas which together will imply the desired result.

Lemma 1. For all σ, there is exactly one σ' such that

$$\langle \mathbf{if}\, X \leq 0\, \mathbf{then}\, Z := Y\, \mathbf{else}\, Z := X + Y, \sigma\rangle \rightarrow_{let} \sigma'$$

specifically,

$$\sigma' = \begin{cases} \sigma[\sigma(Y)/Z] & \text{if } \sigma(X) \leq 0, \\ \sigma[(\sigma(X) + \sigma(Y))/Z] & \text{otherwise.} \end{cases}$$

Proof: Trivial. Just consider the evaluation rules with an arbitrary σ. There are then two cases, one for $\sigma(X) \leq 0$ and one for otherwise. Both are easy. ■

Lemma 2. For all σ, there is exactly one σ' such that

$$\langle \mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1, \sigma\rangle \rightarrow_{let} \sigma'$$

$$\sigma' = \begin{cases} \sigma[\sigma(Y)/Z] & \text{if } \sigma(X) \leq 0, \\ \sigma[(\sigma(X) + \sigma(Y))/Z] & \text{otherwise.} \end{cases}$$

It is then trivial to prove the desired result from the two Lemmas.

We now prove Lemma 2

Proof: It is not possible to do a "mathematical induction" all of the integers, because there is "no place to start." So first we prove the Lemma in a single step for all σ such that $\sigma(X) \leq 0$. Then we prove that the Lemma holds for all σ such that $\sigma(X) \geq 0$ by an induction on $n = \sigma(X)$. Combining these two pieces proves the lemma for all σ which is what we need, as all states σ fall in at least one these two cases (σ such that $\sigma(X) = 0$ falls in both cases, but that is ok!)

So we wish to show that for all σ such that $\sigma(X) \leq 0$ there is exactly one state which $\langle \mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1, \sigma\rangle$ evals to, namely $\sigma[\sigma(Y)/Z]$.

A careful examination of the rules for $\mathbf{IMP}_{let}$ show that there is exactly one derivation, D starting from the configuration $\langle \mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1, \sigma\rangle$ when $\sigma(X) \leq 0$. The derivation looks like:

$$\dfrac{\dfrac{\dfrac{\vdots}{\langle X \leq 0, \sigma\rangle \rightarrow_{let} \mathbf{true}} \qquad D'\left\{\dfrac{\vdots}{\langle Z := Y, \sigma\rangle \rightarrow_{let} \sigma'}\right.}{\langle \mathbf{if}\, X \leq 0\, \mathbf{then}\, Z := Y\, \mathbf{else}(X := X - 1;\ \mathbf{lr}_1;\ X := X + 1;\ Z := Z + 1), \sigma\rangle \rightarrow_{let} \sigma'}}{\langle \mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1, \sigma\rangle \rightarrow_{let} \sigma'}$$

where $\mathbf{lr}_1$ is $\mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1$.

Looking at the subderivation D' of D, we see that there is exactly one possible D', which enforces that $\sigma' = \sigma[\sigma(Y)/Z]$, and so we are done with this case..

Now we use induction on $n = \sigma(X)$ to prove the Lemma for all σ such that $\sigma(X) \geq 0$. Specifically the property $P(n)$ which we are trying establish is defined to be:

> For all states σ such that $\sigma(X) = n$, there is exactly one σ' such that $\langle \mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1, \sigma\rangle \rightarrow_{let} \sigma'$, specifically $\sigma' = \sigma[(n + \sigma(Y))/Z]$.

Basis. $n = 0$. By the preceeding case of the proof ($\sigma(X) \leq 0$) we know that for any σ such that $\sigma(X) = 0$, $\sigma' = \sigma[\sigma(Y)/Z]$. As $n = 0$, $\sigma' = \sigma[(n + \sigma(Y))/X]$.

Inductive step $n = n' + 1$ ($n' \geq 0$). We now consider the form that a derivation D, of $\langle \mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1, \sigma\rangle \rightarrow_{let} \sigma'$, must take.

$$\dfrac{\dfrac{\dfrac{\vdots}{\langle X \leq 0, \sigma\rangle \rightarrow_{let} \mathbf{false}} \qquad \dfrac{\dfrac{\vdots}{\langle X := X - 1, \sigma\rangle \rightarrow_{let} \sigma[n'/X]} \quad D'\left\{\dfrac{\vdots}{\cdots}\right.}{\langle X := X - 1;\ \mathbf{lr}_1;\ X := X + 1;\ Z := Z + 1, \sigma\rangle \rightarrow_{let} \sigma'}}{\langle \mathbf{if}\, X \leq 0\, \mathbf{then}\, Z := Y\, \mathbf{else}(X := X - 1;\ \mathbf{lr}_1;\ X := X + 1;\ Z := Z + 1), \sigma\rangle \rightarrow_{let} \sigma'}}{\langle \mathbf{letrec}\, add\, \mathbf{in}\, \mathbf{p}_1, \sigma\rangle \rightarrow_{let} \sigma'}$$

Where D' is must exist and have the form:

$$\frac{\frac{\vdots}{\langle \mathbf{lr}_1, \sigma[n'/X]\rangle \rightarrow_{let} \sigma''} \quad \frac{\vdots}{\langle X := X+1;\ Z := Z+1, \sigma''\rangle \rightarrow_{let} \sigma'}}{\langle \mathbf{lr}_1;\ X := X+1;\ Z := Z+1, \sigma[n'/X]\rangle \rightarrow_{let} \sigma'}$$

Examining the form of D, we see that it exists iff we have a derivation D' for $\langle \mathbf{lr}_1, \sigma[n'/X]\rangle \rightarrow_{let} \sigma''$. We do. Moreover, once σ'' is chosen there remains exactly one σ' which allows this derivation to exist, moreover there will always be such a σ', forcing this derivation to exist. Specifically, $\sigma' = \sigma''[(1+\sigma''(X)/X][(1+\sigma''(Z))/Z]$.

By induction, there was exactly one σ'' such that $\langle \mathbf{lr}_1, \sigma[n'/X]\rangle \rightarrow_{let} \sigma''$, namely $\sigma'' = \sigma[n'/X][(n'+\sigma(Y))/Z]$. Consequently, there is exactly one σ' such that $\langle \mathbf{letrec}\ add\ \mathbf{in}\ \mathbf{p}_1, \sigma\rangle \rightarrow_{let} \sigma'$, specifically,

$$\sigma[n'/X][(n'+\sigma(Y)/Z][(1+n')/X][(1+n'+\sigma(Y))/Z] = \sigma[(n'+\sigma(Y))/Z]$$

exactly as required. ■

1(b). What can go wrong if X, Y, Z happen not be distinct?

Solution: A lot. Consider, for example if X and Z are the same. Then we will end up with the end state having $\sigma(X) \cdot 2 + \sigma(Y)$ in location X. If X and Y coincide, you end up with Z getting 0. Or, if Y and Z coincide, nothing bad happens. If all 3 coincide then you end up with 2 times the original value in that location.

Summary: for this particular piece of code problems only arise when X is the same **Loc** as Y or X is the same **Loc** as Z (but not when all three are the same **Loc**).

Note any violation of the distinctness property violates the soundness of the proof of Lemma 2.

Any combination of problems, or an abstract discussion of where the proof breaks down, will be accepted with full credit.

Problem 2.

2(a). Describe how to translate any $\mathbf{IMP}_{let}$ command into an equivalent $\mathbf{IMP}_{let}$ command which does not contain any **while** constructs. *Hint*: Show how to eliminate an "innermost" **while**.

Solution: We show how to convert an $\mathbf{IMP}_{let}$ command containing a **while** structure into a new $\mathbf{IMP}_{let}$ which contains one fewer **while** structure. It would then be a trivial induction on the number of while structures to show that repeated application of this conversion is guaranteed to reach an $\mathbf{IMP}_{let}$ command which does not contain any **while** structures.

We do this by showing how to eliminate an "innermost" **while** from an $\mathbf{IMP}_{let}$ command. (An "innermost" **while** is a subpart of a command which is itself a **while** structure, but which has no **while** structures contained within it). We have two cases depending on whether the innermost **while** structure is closed or open.

Without loss of generality, we may assume that there is a single innermost **while** structure (if in fact there are several, we can just pick an arbitrary one, and call it "the innermost" **while** structure).

If the innermost **while** structure is closed, then it is of the form $\mathbf{while}\, b\, \mathbf{do}\, c$ where both b and c are closed. We can replace it by:

$$\mathbf{letrec\, if}\, b\, \mathbf{then}(c; \mathbf{p}_1)\, \mathbf{else\, skip\, in}\, \mathbf{p}_1$$

If the innermost **while** structure is open, then we need to look at the enclosing **letrec** structure. Suppose this enclosing **letrec** is of the form:

$$\mathbf{letrec}\, \vec{p_n}\, \mathbf{in}\, c$$

Let the **while** structure be $\mathbf{while}\, b\, \mathbf{do}\, c_0$. We then extend the vector of $\vec{p_n}$'s by $\mathbf{if}\, b\, \mathbf{then}(c_0; \mathbf{p}_{n+1})\, \mathbf{else\, skip}$ and we replace the **while** structure by $\mathbf{p}_{n+1}$.

In other words, we end up with:

$$\mathbf{letrec}\, \vec{p_n}\,', \mathbf{if}\, b\, \mathbf{then}(c_0; \mathbf{p}_{n+1})\, \mathbf{else\, skip\, in}\, c'$$

Where, if the **while** structure which we were eliminating was in c, then c' is c with the **while** structure replaced by $\mathbf{p}_{n+1}$, and $\vec{p_n}\,' \equiv \vec{p_n}$. Otherwise we were replacing the **while** structure in p_i for some i. In which case $c' \equiv c$, and $p'_j \equiv p_j$(for $1 \leq j \leq n$ and $j \neq i$), and p'_i is p_i with the **while** structure replaced by $\mathbf{p}_{n+1}$.

2(b). Illustrate your method by translating

$$\begin{array}{l}
\mathbf{letrec}\ \mathbf{while}\, b_1\, \mathbf{do}\ (X := X + 1; \mathbf{p}_2),\\
\qquad Y := Y + 1; \mathbf{p}_1\\
\mathbf{in}\\
\qquad \mathbf{while}\ b_2\ \mathbf{do}\\
\qquad\quad \mathbf{p}_1;\\
\qquad\quad \mathbf{letrec}\, Y := Y - 1\\
\qquad\quad \mathbf{in} \qquad \mathbf{while}\, b_3\, \mathbf{do}\, \mathbf{p}_1
\end{array}$$

Solution:

letrec $\mathbf{p_3}$
$Y := Y + 1; \mathbf{p_1}$,
if b_1 **then** $(X := X + 1; \mathbf{p_2}); \mathbf{p_3}$ **else skip**,
if b_2 **then** $\mathbf{p_1}$;
letrec $Y := Y - 1$,
if b_3 **then**$(\mathbf{p_1}; \mathbf{p_2})$ **else skip**
in $\mathbf{p_2}$
else skip
in $\mathbf{p_4}$

Revised Problem 3 Γ_p can be defined by an induction on the structure of **OCom**. Setting up this induction has some subtle points. In particular consider the case of $p \equiv \mathbf{while}\, b\, \mathbf{do}\, p_0$. For this case we have:

$$\Gamma_{\mathbf{while}\, b\, \mathbf{do}\, p_0}(\vec{\varphi}_n) = fix(\Gamma'),$$

where Γ' is defined by:

$$\Gamma'(\psi)\sigma = \begin{cases} \sigma & \text{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{false}, \\ \psi(\Gamma_{p_0}(\vec{\varphi}_n)\sigma) & \text{otherwise.} \end{cases}$$

The goal of this problem is to show that $\Gamma_{\mathbf{while}\, b\, \mathbf{do}\, p_0}$ is well-defined (assuming that Γ_{p_0}) is well defined. To do so we show that $fix(\Gamma')$ is well-defined.

3(a). Prove that $\Gamma' : (\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$ is continuous. *i.e.* show that if $\psi_0, \psi_1, \ldots$ is an ascending chain in $\Sigma \rightharpoonup \Sigma$, then

$$\Gamma'\left(\bigsqcup_{n\geq 0} \psi_n\right) = \bigsqcup_{n\geq 0} (\Gamma'(\psi_n))$$

Recall that we order $\Sigma \rightharpoonup \Sigma$ under $\subseteq$.

Solution. To show that two partial functions $f, g : A \rightharpoonup E$ are equal, it is necessary to show that for all $d \in D$,

either $f(d)$, and $g(d)$ are both undefined,
or $f(d)$ and $g(d)$ are both defined, and have the same value.

So, to show that Γ' is continuous, we must show that for an arbitrary ascending chain in $\Sigma \rightharpoonup \Sigma$ (say, $\psi_1, \psi_2, \ldots$), and all states σ:

either $\Gamma'(\bigsqcup_{n\geq 0}\psi_n)(\sigma)$, and $(\bigsqcup_{n\geq 0}\Gamma'(\psi_n))(\sigma)$ are both undefined,
or they are both defined, and have the same value.

We prove this by cases on $\mathcal{B}[\![b]\!]\sigma$.

If $\mathcal{B}[\![b]\!]\sigma = \textbf{false}$, then both functions at σ give σ. Why? By the definition of Γ', $\Gamma'(\textit{anything})\sigma = \sigma$. So, $\Gamma'(\bigsqcup_{n\geq 0}\psi_n)\sigma = \sigma$. Similarly, for all n, $\Gamma(\psi_n)\sigma = \sigma$. Thus $\bigsqcup_{n\geq 0}(\Gamma'(\psi_n)) = \bigsqcup\{\sigma\} = \sigma$, and so we are done.

If it is not the case that $\mathcal{B}[\![b]\!]\sigma = \textbf{false}$ then,

$$\Gamma'\left(\bigsqcup_{n\geq 0}\psi_n\right)\sigma = \left(\bigsqcup_{n\geq 0}\psi_n\right)(\Gamma_{p_0}(\vec{\varphi}_n)\,\sigma)$$

moreover, by the definition of $\sqsubseteq_{(\Sigma\rightarrow\Sigma)\rightarrow(\Sigma\rightarrow\Sigma)}$

$$\begin{aligned}\left(\bigsqcup_{n\geq 0}\Gamma'(\psi_n)\right)\sigma &= \left(\bigsqcup_{n\geq 0}\Gamma'(\psi_n)\sigma\right)\\ &= \bigsqcup_{n\geq 0}(\psi_n(\Gamma_{p_0}\sigma))\end{aligned}$$

The last step came from the definition of Γ'. Let $\sigma' = \Gamma_{p_0}(\vec{\varphi}_n)\sigma$, if it is defined, (if it is not defined then we are done—as "$f(\sigma)$" and "$g(\sigma)$" both end up being undefined).

So, if we can prove the following statement, we will be done.

$$\left(\bigsqcup_{n\geq 0}\psi_n\right)\sigma' = \bigsqcup_{n\geq 0}(\psi_n)\,\sigma'$$

We can in fact prove the stronger statement:

$$\left(\bigsqcup_{n\geq 0}\psi_n\right)(\Gamma_{p_0}(\vec{\varphi}_n)) = \bigsqcup_{n\geq 0}\Big(\psi_n(\Gamma_{p_0}(\vec{\varphi}_n))\Big)$$

which comes directly from the definition of $\sqsubseteq_{\Sigma\rightarrow\Sigma}$.

Actually, this case has not properly worried about the possibility that one of two sides of an equation might be undefined. This can be done by replacing certain occurrences of $=$ by $\simeq$, which stands for: the two things on both sides, are either both undefined, or they are both equal. Don't worry about this for now.

3(b). Show that Γ' has a least fixpoint, and thus conclude that $fix(\Gamma')$ is well defined.

Solution. In part (a) we have shown that $\Gamma' : (\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$ is continuous. We mentioned in class that $\Sigma \rightharpoonup \Sigma$ ordered under $\subseteq$ is a cpo. Moreover, this cpo has a bottom (*viz.* least element). Specifically the least element of $\Sigma \rightharpoonup \Sigma$ is the completely undefined function (the function whose graph is the empty set). This enables us to apply Theorem 16 from page 64 of Winskel, to say that Γ' has a least fixpoint. And since Γ' is well-defined, and Γ' has a least fixpoint, $fix(\Gamma')$ is well-defined.

Revised Problem 3-OPT. (OPTIONAL) Do the original Problem 3 from Problem Set 5.

Carefully define Γ_p. *Hint*: Structural induction on **OCom**.

Solution: We define Γ_p by induction on the structure of **OCom**. There are some subtle problems involved in properly setting up the induction and really showing that Γ_p is well defined. The main problem arises from the fact that defining Γ_p involves $\mathcal{C}[\![\cdot]\!]$, which in turn would involve Γ_p because of the **letrec** construct. Some additional cleverness (which is not worth going into here) actually justifies the straightforward definition which follows. There is also some work involved in showing that Γ_p was well-defined. What was assigned as Problem 3 in the addendum, shows what is necessary to show that $\Gamma_{\mathbf{while}\, b\, \mathbf{do}\, c}$ is well-defined, one of the harder cases.

There are five, top level cases based on the structure of p.

The base cases are:

[$p \equiv \mathbf{p}_i$] All of the complex definitions we have set up, and the whole structure of this problem was designed to make this case work out easily. The rest of the work is then to show that the other cases still come out ok. For this case, we have:

$$\Gamma_{\mathbf{p}_i}(\varphi_1, \ldots, \varphi_n) \stackrel{\text{def}}{=} \varphi_i$$

[$p \equiv c$] For a closed command we simply ignore $\vec{\varphi}_n$. Giving,

$$\Gamma_c(\vec{\varphi}_n) \stackrel{\text{def}}{=} \mathcal{C}[\![c]\!]$$

The inductive cases are:

[$p \equiv p_0; p_1$] This is simply a composition, we just need to be careful where we do it.

$$\Gamma_{p_1;p_2}(\vec{\varphi}_n) \stackrel{\text{def}}{=} (\Gamma_{p_2}(\vec{\varphi}_n)) \circ (\Gamma_{p_1}(\vec{\varphi}_n))$$

[$p \equiv$ **if** b **then** p_0 **else** p_1] This is not too hard either...

$$\Gamma_{\mathbf{if}\,b\,\mathbf{then}\,p_0\,\mathbf{else}\,p_1}(\vec{\varphi}_n) \stackrel{\text{def}}{=} \begin{cases} \Gamma_{p_0}(\vec{\varphi}_n)\sigma & \text{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{true}, \\ \Gamma_{p_1}(\vec{\varphi}_n)\sigma & \text{otherwise.} \end{cases}$$

[$p \equiv$ **while** b **do** p_0] For **while** structures, we still need to define an auxiliary function and find the fixpoint of that function. So, we have

$$\Gamma_{\mathbf{while}\,b\,\mathbf{do}\,p_0}(\vec{\varphi}_n) \stackrel{\text{def}}{=} \mathit{fix}(\Gamma')$$

Where

$$\Gamma'(\psi)\sigma \stackrel{\text{def}}{=} \begin{cases} \sigma & \text{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{false}, \\ \psi(\Gamma_{p_0}(\vec{\varphi}_n)\sigma) & \text{otherwise.} \end{cases}$$

Notice that Γ' depends on $\vec{\varphi}_n$.

Revised Problem 4-OPT. (OPTIONAL) Do the original Problem 4 from Problem Set 5.

4-OPT(a). Show that Γ_p is continuous.

Solution: There are too many separate pieces involved in showing Γ_p continuous, so at this point, we are not writing up official solutions to the problem. Problem 3 (the new version), contains the hardest case, which is in fact a combination of several of the tricks needed.

4-OPT(b). Conclude that for well-formed $\vec{p}_n$, the function $\Gamma_{\vec{p}_n}$ is continuous.

Solution: This follows directly from a generalization of the fact given in class:

Fact 1. A function $f : A \times B \to C$ is continuous iff

$$\begin{array}{rll} & f_a & \text{is continuous for all } a \in A, \\ \text{and} & f^b & \text{is continuous for all } b \in B. \end{array}$$

In English, the generalization is that a function in n arguments is continuous iff it is continuous in each of its arguments separately. *e.g.* f with all but its fifth argument fixed at arbitrary values, considered as a function in just that fifth argument, is continuous. If this happens for all n argument slots, then f is continuous. Obviously $\Gamma_{\vec{p}_n}$ will fit this bill!!

Problem Set 7

Reading assignment. Winskel Chapter §7.1- §7.3.

Quiz 3 Room Announcement: Quiz 3, which the class unanimously decided would take place on Tuesday November 19, from 7 to 9pm, will be held in **Room 34-301**, across the hall from the usual classroom.

Due: 15 November 1991.

Problem 1. Let $loc(B)$ be the set of locations which occur in a formula $B \in$ **Assn**.

Let $\mathcal{L}$ be a subset of **Loc**. We define an equivalence relation on states, $\sim_{\mathcal{L}}$, as follows:

$$\sigma_1 \sim_{\mathcal{L}} \sigma_2 \text{ iff } \forall X \in \mathcal{L}. \sigma_1(X) = \sigma_2(X).$$

In other words, $\sigma_1 \sim_{\mathcal{L}} \sigma_2$ iff the states σ_1 and σ_2 agree on all locations in $\mathcal{L}$.

1(a). Prove that an assertion depends only on locations explicitly mentioned in it, that is:

$$\text{if } \sigma_1 \sim_{loc(B)} \sigma_2 \text{ then } (\sigma_1 \models^I B \text{ iff } \sigma_2 \models^I B).$$

Hint: Use structural induction on B.

1(b). Using part (a), give a direct semantic proof that

$$\text{if } loc(B) \cap loc_L(c) = \emptyset, \text{ then } \models \{B\}c\{B\}.$$

(Do not appeal to soundness or completeness of the Hoare rules.)

Hint: Winskel Proposition 8, p. 45.

Problem 2. In this problem, you will give a syntactic proof of the result of the preceding problem. Specifically, we want you to show that:

$$\text{if } loc(B) \cap loc(c) = \emptyset \text{ then } \vdash_{\text{Hoare}} \{B\}c\{B\}$$

Prove this by structural induction on c. Do so directly from the definition of the Hoare rules and axioms. (Do not appeal to the Completeness Theorem or the result of Problem 1).

Problem 3. In class we gave the following definition of the **DynAssn** abbreviation for the weakest precondition under c for $d \in$ **DynAssn**:

$$w(c, D) ::= \{\mathbf{true}\}c\{D\}$$

Note we used a small "w" in this definition. This is not to be confused with $W(c, B) \in$ **Assn**, which we defined in class for $B \in$ **Assn** (although $w(c, B)$ is equivalent to $W(c, B)$).

Prove from the definition of validity for **DynAssn**:

$$\models w((c_1; c_2), D) \Rightarrow w(c_1, w(c_2, D))$$

(Of course the converse ($\Leftarrow$) also holds, but we thought that it was enough of an exercise to prove the equivalence in one direction)

Problem 4. An important technical fact for demonstrating the expressiveness of **Assn** comes from demonstrating that you can code sequences in **Assn**. In class we assumed there was a formula $SEQ \in$ **Assn**. Winskel describes the β-function, which is one way of implementing SEQ in **Assn**.

Instead of building up SEQ from some ingenious number theory (which is what β requires), we can code up SEQ by "string manipulation." As a first step, we observe that a **Num** can be represented as a sequence of characters (via their representation in some particular base, say base 3). We expect that everyone knows how to write a positive integer as its base 3 representation (which is some particular sequence of "0"'s, "1"'s and "2"'s). We could then perform some "string" operations on the base three representations, and then finally convert the end string s to the number which has s as its base 3 representation. It is easy to code a sequence of strings as one long string by concatenating the strings in the sequence, separated say by delimiter symbols. We do not have time to go all the way up to coding SEQ, but will just work out how to define string concatenation.

We show to think of numbers as strings and how to represent concatenation of two strings. For this problem we will use the following notation: for $n \geq 0$ we write $(n)_3$ to denote the base 3 representation of n (without leading zeros). In our informal discussion, we will use $\cdot$ do denote the concatenation of strings, for example "aab"·"bbba"="aabbbba"

For a more concrete example of what will go on in this problem:

$$\begin{aligned}
(5)_3 &= \text{"12"} \\
(21)_3 &= \text{"210"} \\
\\
(5)_3 \cdot (21)_3 &= \text{"12210"} \\
&= (3^4 + 2 \times 3^3 + 2 \times 3^2 + 3)_3 \\
&= (156)_3
\end{aligned}$$

Concretely, in this problem we will guide you to defining a formula $F \in$ **Assn** with free variables i, j, k such that F means: "$(i)_3 \cdot (j)_3 = (k)_3$".

4(a). Define a formula $POW3(i) \in$ **Assn**, which is true iff i is a power of 3.

Hint: i is a power of 3 iff any divisor of i other than 1 must itself have 3 as a divisor.

4(b). In our informal language, let length(i), for $i \geq 0$ be the length of the base 3 representation of i. Find a formula $LEN(i, j) \in$ **Assn**, such that LEN is true iff $j = 3^{\text{length}(i)}$.

Hint: j is the largest power of 3 with a certain relation to i.

4(c). Define a formula $F(i, j, k) \in$ **Assn**, which is true iff $(k)_3 = (i)_3 \cdot (j)_3$.

Hint: $k = 3^{\text{length}(j)} \times i + j$.

Notes on Expressiveness

Lemma 1. There is a formula SEQ $\in$ Assn which means "i is the code of a sequence whose j^{th} element is k."

Proof: Ingenious specification using elementary number theory. Winskel gets such a formula using Gödel's "β" function. Another approach is hinted at in Problem Set 7. ■

With a SEQ formula we can obtain "pairing" as a special case:

$$\begin{aligned} \text{LEFT} &::= \text{SEQ}[0/j] \\ \text{RIGHT} &::= \text{SEQ}[1/j] \end{aligned}$$

So LEFT means "i is the code of a pair whose left element is k" and similarly for RIGHT.

We will want to use other integer variables besides i, j, k in formulas, so we write

$$\text{SEQ}(i', j', k')$$

as an abbreviation for $\text{SEQ}[i'/i][j'/j][k'/k]$. Note that Winskel does not define what it means to substitute an **Aexp** *with integer variables* for an integer variable, so technically we need to define $A[i'/i]$. However, as long as i' is "fresh," namely there are no occurrences (free *or* bound) of i' in A, then the definition used for $A[a/i]$ also is ok for $A[i'/i]$, and the formula $\text{SEQ}(i', j', k')$ will mean, as expected, that "i' is the code of a sequence whose j'^{th} element is k'." Likewise for $\text{LEFT}(i', k')$, etc.

(To illustrate the technical problem with substitution of integer variables which are not fresh, consider the formula $A ::= (\exists i'. 2 \times i' = i)$ which means "i is even." Now the naive definition of $A[i'/i]$ yields the formula $A ::= (\exists i'. 2 \times i' = i')$ which happens to be *valid*, and so certainly does not mean "i' is even.")

We now show how to construct a formula $W(c, B) \in$ **Assn** expressing the weakest precondition of c for any $B \in$ **Assn** where $c \equiv$ **while** b **do** c_0. By structural induction we have formulas $W(c_0, B')$ for all $B' \in$ **Assn**.

For notational simplicity, assume $\text{Loc}(B) \cup \text{Loc}(c) = \{X_1, X_2\}$.

For formulas and commands with only locations $\{X_1, X_2\}$, the only part of a state which is relevent for satisfaction or command meaning are the pair of values of X_1 and X_2, so we "code" a state σ as a number n such that $\text{LEFT}(n, n_1)$ and $\text{RIGHT}(n, n_2)$ are both valid, where $n_1 = \sigma(X_1)$ and $n_2 = \sigma(X_2)$. Conversely, let *state*(n) be a state, σ, such that $\sigma(X_1) = n_1$ and $\sigma(X_2) = n_2$ for the (necessarily unique) numbers n_1, n_2 such that $\models \text{LEFT}(n, n_1) \wedge \text{RIGHT}(n, n_2)$. For definiteness, we may assume $(state(n))(Y) = 0$ for $Y \not\equiv X_1, X_2$.

The formula $\text{SAT}(k, B')$ will mean "$state(k) \models B'$" where $B' \in$ **Assn** and $\text{Loc}(B') \subseteq \{X_1, X_2\}$.

$$\text{SAT}(k, B') ::= \exists k_1.\exists k_2.\, \text{LEFT}(k, k_1) \wedge \text{RIGHT}(k, k_2) \wedge B'[k_1/X_1][k_2/X_2].$$

(Note that both k_1 and k_2 have to be "fresh" integer variables for SAT to work properly.)

Our explanation of the meaning of SAT has been informal about the role of the interpretation, I. More precisely, we should have said that

$$\sigma \models^I \text{SAT}(k, B') \quad \text{iff} \quad state(I(k)) \models^I B'.$$

Notice that the truth or falsehood of $\text{SAT}(k, B')$ depends only on the interpretation I, not the state σ, since $\text{SAT}(k, B')$ does not contain any locations.

Another useful formula is $\text{PS}(k)$ which means "the present state has code k."

$$\text{PS}(k) ::= \text{LEFT}(k, X_1) \wedge \text{RIGHT}(k, X_2)$$

So $\sigma \models^I \text{PS}(k)$ iff $[\sigma(X_i) = state(I(k))(X_i)$ for $i = 1, 2]$.

The formula $\text{NEXT}(c_0, l_3, l_4)$ means "$\mathcal{C}[\![c_0]\!](state(l_3)) = state(l_4)$"

$$\text{NEXT}(c_0, l_3, l_4) ::= \text{SAT}(l_3, W(c_0, \text{PS}(l_4))) \wedge \text{SAT}(l_3, \neg W(c_0, false))$$

Exercise. Why is the second conjunct needed in the definition of NEXT?

Now we can define $W(c, B)$ as follows. We use n as the length of a sequence $\sigma_0, \sigma_1, \ldots, \sigma_n$ of states, i as the code for the sequence of numbers coding these states, l_1 as the code for σ_0, and l_2 as the code for σ_n.

Then $\sigma_0 \models W(c, B)$ iff

$$\{\sigma_i \models b \text{ and } \mathcal{C}[\![\sigma_j]\!] = \sigma_{j+1} \text{ for } i \leq j < n, \text{ and } \sigma_n \models \neg b\} \quad \text{implies} \quad \sigma_n \models B.$$

So, $W(c, B) ::= \forall n.\forall i.\forall l_1.\forall l2.$

$\{n \geq 0 \wedge$	
$\text{SEQ}(i, 0, l_1) \wedge$	"$\sigma_0 = state(l_1)$"
$\text{SEQ}(i, n, l_2) \wedge$	"$\sigma_n = state(l_2)$"
$\text{PS}(l_1) \wedge$	"σ_0 is the present state"
$[\forall j.(0 \leq j \wedge \neg(n \leq j)) \Rightarrow$	
$\quad \exists l_3.\exists l_4.(\text{SEQ}(i, j, l_3) \wedge$	"$\sigma_j = state(l_3)$"
$\quad\quad \text{SEQ}(i, j+1, l_4) \wedge$	"$\sigma_{j+1} = state(l_4)$"
$\quad\quad \text{SAT}(l_3, b) \wedge$	"$\sigma_j \models b$"
$\quad\quad \text{NEXT}(c_0, l_3, l_4))] \wedge$	"$\mathcal{C}[\![\sigma_j]\!] = \sigma_{j+1}$"
$\quad \text{SAT}(l_2, \neg b)\} \Rightarrow$	"$\sigma_n \models \neg b$"
$\text{SAT}(l_2, B)$	"$\sigma_n \models B$"

Problem Set 6 Solutions

1 General Information

This handout includes some of the best solutions submitted by students for Problem Set 6. These solutions are a good representation of the level of detail expected.

2 Grades

Attach to each graded solution is a printout of our current record for each student's grades. Please check that all of the raw data is accurate—inform the TA for corrections. In addition we have ranked everyone's quiz scores (from 1 to 17, 1=best), and everyone's homework totals. This information is included on the printout.

For your convenience, the following is a summary of the grade statistics to this date:

			PS1	PS1	PS1	PS2	PS2	PS3	PS3	PS3	PS4	PS4
	Quiz1	Quiz2	Prob0	Prob1	Prob2	Prob1	Prob2	prob1	Prob2	Prob3	Prob1	Prob2
# Submitte	17	16	16	16	16	13	13	16	16	16	12	14
High	66	95	10	10	10	8	20	10	10	10	7	10
Low	23	28	8	1	1	3	1	0	1	4	1	3
Median	45.0	62.5	10.0	3.0	4.0	5.0	7.0	9.0	8.0	10.0	2.50	7.50
Mean	46.2	62.1	9.8	4.2	5.2	5.3	9.2	8.3	7.7	8.6	3.25	7.36
St. Dev.	14.7	20.4	0.6	3.3	3.3	1.8	6.4	2.6	2.2	1.9	1.76	1.74

		PS4	PS4	PS5	PS5	PS5	PS6	PS6	PS6	HW
		Prob3	Prob4	Prob1	Prob2	Prob3	Prob1	Prob2	Prob3	Tots
	# Submitte	14	14	14	14	8	15	15	15	17
	High	10	10	10	10	5	10	10	10	158
	Low	2	5	3	2	0	3	6	5	27
	Median	4.00	7.00	8.00	3.00	3.00	9.00	9.00	8.00	104.00
	Mean	4.71	7.21	7.64	4.71	2.63	7.73	8.53	7.80	103.00
	St. Dev.	2.76	1.72	2.13	2.87	1.60	2.46	1.25	1.47	37.32

10/10

Pete Rauch
6.044/1[illegible]23
PS# 6
11-8-1991

Problem 1

1(a) Prove $\{V \geq 1 \text{ and } Z \times W^V = W'^{V'}\}\ B\ \{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$

By two uses of the sequencing rule, we must show

1. $\{V \geq 1 \text{ and } Z \times W^V = W'^{V'}\}\ X := V\ \{C\}$
2. $\{C\}\ D\ \{E\}$
3. $\{E\}$ if $X=0$ then $W := W \times W; V := Y$ else $Z := Z \times W; V := V-1$ $\{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$

$C \equiv (V \geq 1 \text{ and } Z \times W^V = W'^{V'} \text{ and } X \geq 1 \text{ and } X = V)$
$E \equiv (V \geq 1 \text{ and } Z \times W^V = W'^{V'} \text{ and } Y = \lfloor V/2 \rfloor \text{ and } X = rem(V,2))$

1. Rule of assignment: $\{C[V/X]\}\ X := V\ \{C\}$, so
$\{V \geq 1 \text{ and } Z \times W^V = W'^{V'} \text{ and } V \geq 1 \text{ and } V = V\}\ X := V\ \{C\}$.
$V \geq 1 \text{ and } Z \times W^V = W'^{V'} \Rightarrow V \geq 1 \text{ and } Z \times W^V = W'^{V'} \text{ and } V \geq 1 \text{ and } V = V$
so by rule of consequence, $\{V \geq 1 \text{ and } Z \times W^V = W'^{V'}\}\ X := V\ \{C\}$ ✓

2. We know $\{X = X' \text{ and } X \geq 0\}\ D\ \{Y = \lfloor X'/2 \rfloor \text{ and } X = rem(X', 2)\}$
for $X' \notin loc(D)$. $(loc(D) = \{X, Y\})$.
Since $V, W, V', W', Z \notin loc(D)$ and $C \Rightarrow X = V$ and $X \geq 0$,
by pattern matching
$\{V \geq 1 \text{ and } Z \times W^V = W'^{V'} \text{ and } X \geq 1 \text{ and } X = V\}\ D\ \{V \geq 1 \text{ and } Z \times W^V = W'^{V'} \text{ and } Y = \lfloor V/2 \rfloor$
$\text{and } X = rem(V,2)\}$ ✓

3. Using the conditional rule we show two cases:

V was even 1. $\{E \wedge X=0\}\ W := W \times W; V := Y\ \{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$

V was odd 2. $\{E \wedge \neg(X=0)\}\ Z := Z \times W; V := V-1\ \{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$

Pete Rauch
6.044/18.423
PS#6
11-8-1991

1(a) (cont.)

3. case 1. By using the sequencing rule, we must show

i $\{E \wedge X=0\}\ W:=W\times W\ \{F_1\}$

ii $\{F_1\}\ V:=Y\ \{V\geq 0 \text{ and } Z\times W^V = w'^{v'}\}$

$F_1 \equiv (Y\geq 0 \text{ and } Z\times W^Y = w'^{v'})$

$V\geq 1 \text{ and } Z\times W^V = w'^{v'} \text{ and } Y=\lfloor V/2\rfloor \text{ and } X=rem(V,2) \text{ and } X=0 \Rightarrow Y=V/2 \text{ and } V\geq 1 \text{ and } Z\times W^V = w'^{v'}$
$\Rightarrow Y=V/2 \text{ and } V\geq 1 \text{ and } Y\geq 1 \text{ and } Z\times W^V = w'^{v'} \Rightarrow Y\geq 0 \text{ and } Z\times W^{2\times Y} = w'^{v'} \text{ and } Y=V/2$

ii Rule of assignment: $\{Y\geq 0 \text{ and } Z\times W^Y = w'^{v'}\}\ V:=Y\ \{V\geq 0 \text{ and } Z\times W^V = w'^{v'}\}$

i Rule of assignment: $\{Y\geq 0 \text{ and } Z\times (W\times W)^Y = w'^{v'}\}\ W:=W\times W\ \{Y\geq 0 \text{ and } Z\times W^Y = w'^{v'}\}$
$Y=V/2 \text{ and } Y\geq 0 \text{ and } Z\times W^{2(V/2)} = w'^{v'} \Rightarrow Y\geq 0 \text{ and } Z\times(W\times W)^Y = w'^{v'}$

So by the rule of consequence and two rules of assignment,
$\{E\wedge X=0\}\ W:=W\times W;\ V:=Y\ \{V\geq 0 \text{ and } Z\times W^V = w'^{v'}\}$

3. case 2. For sequencing, we need

i $\{E\wedge \neg(X=0)\}\ Z:=Z\times W\ \{F_2\}$

ii $\{F_2\}\ V:=V-1\ \{V\geq 0 \text{ and } Z\times W^V = w'^{v'}\}$

$F_2 = (V\geq 1 \text{ and } Z\times W^{V-1} = w'^{v'})$

Assignment ii $\{V\geq 1 \text{ and } Z\times W^{V-1} = w'^{v'}\}\ V:=V-1\ \{V\geq 0 \text{ and } Z\times W^V = w'^{v'}\}$

Assignment i $\{V\geq 1 \text{ and } Z\times W\times W^{V-1} = w'^{v'}\}\ Z:=Z\times W\ \{V\geq 1 \text{ and } Z\times W^{V-1} = w'^{v'}\}$

$V\geq 1 \text{ and } Z\times W^V = w'^{v'} \text{ and } Y=\lfloor V/2\rfloor \text{ and } X=rem(V,2) \text{ and } \neg(X=0)$
$\Rightarrow V\geq 1 \text{ and } Z\times W^1\times W^{V-1} = Z\times W\times W^{V-1} = Z\times W^V = w'^{v'}$

So by consequence and two assignments,

$\{E\wedge\neg(X=0)\}\ Z:=Z\times W;\ V:=V-1\ \{V\geq 0 \text{ and } Z\times W^V = w'^{v'}\}$

We have shown 1., 2., & 3., so $\{V\geq 1 \text{ and } Z\times W^V = w'^{v'}\}\ B\ \{V\geq 0 \text{ and } Z\times W^V = w'^{v'}\}$

✓

Pete Rauch
6.044/[illegible]3
PS#6
11-8-1991

1(b) Conclude that:

$\{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$ while $\neg(V \leq 0)$ do B $\{V = 0 \text{ and } Z = W'^{V'}\}$

from 1(a) we know that

$\{V \geq 1 \text{ and } Z \times W^V = W'^{V'}\}$ B $\{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$

$\neg(V \leq 0)$ and $V \geq 0$ and $Z \times W^V = W'^{V'} \Rightarrow V \geq 1$ and $Z \times W^V = W'^{V'}$

so by the rule of consequence and 1(a) we have:

$\{\neg(V \leq 0) \text{ and } V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$ B $\{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$

Using the rule for while loops we derive

$\{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$ while $\neg(V \leq 0)$ do B $\{(V \leq 0) \text{ and } V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$

$(V \leq 0)$ and $(V \geq 0)$ and $Z \times W^V = W'^{V'} \Rightarrow V = 0$ and $Z \times W^V = W'^{V'}$
$\Rightarrow V = 0$ and $Z \times 1 = W'^{V'} \Rightarrow V = 0$ and $Z = W'^{V'}$

So, by the rule of consequence we conclude:

$\{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$ while $\neg(V \leq 0)$ do B $\{V = 0 \text{ and } Z = W'^{V'}\}$.

Pete Rauch
6.044/18.423
PS#6
11-8-1991

1(c) Conclude that:

$\{V \geq 1 \text{ and } V' = V \text{ and } W' = W\}$ Exp $\{W'^{V'} = Z\}$

Using the sequencing rule on Exp, we must show:

1. $\{V \geq 1 \text{ and } V' = V \text{ and } W' = W\}\ Z := 1\ \{C\}$ & 2. $\{C\}$ while $\neg(V \leq 0)$ do B $\{W'^{V'} = Z\}$

$C \equiv (V \geq 0 \text{ and } Z \times W^V = W'^{V'})$

1. By the assignment rule, $\{V \geq 0 \text{ and } 1 \times W^V = W'^{V'}\}\ Z := 1\ \{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$

$V \geq 1$ and $V' = V$ and $W' = W \Rightarrow V \geq 0$ and $W'^{V'} = W^V \Rightarrow V \geq 0$ and $1 \times W^V = W'^{V'}$,

so by the rule of consequence,

$\{V \geq 1 \text{ and } V' = V \text{ and } W' = W\}\ Z := 1\ \{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$ ✓

2. From 1(b): $\{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$ while $\neg(V \leq 0)$ do B $\{V = 0 \text{ and } Z = W'^{V'}\}$

$V = 0$ and $Z = W'^{V'} \Rightarrow W'^{V'} = Z$, so by the rule of consequence,

$\{V \geq 0 \text{ and } Z \times W^V = W'^{V'}\}$ while $\neg(V \leq 0)$ do B $\{W'^{V'} = Z\}$.

Having shown 1 and 2, we conclude that:

$\boxed{\{V \geq 1 \text{ and } V' = V \text{ and } W' = W\} \text{ Exp } \{W'^{V'} = Z\}}$.

10/10

Mark Haseltine
6.044J / 18.423J
Problem Set 6
11/8/91

Problem 2

Prove $\{true\}$ while true do $X := X+1$ $\{false\}$

Hint: Try the invariant "true".

So, $I \equiv (true)$

Now show I is an ivariant. Take

$$\{I \wedge true\}\ X := X+1\ \{I\}$$

From the rule of assignment

$$\{I[\tfrac{X+1}{X}]\}\ X := X+1\ \{I\}$$

where $I[\frac{X+1}{X}] \equiv I \equiv true$

Clearly $I \wedge true \Rightarrow true$ since $true \wedge true \Rightarrow true$

Thus, by the consequence rule $\{I \wedge true\}\ X := X+1\ \{I\}$

So, I is an invariant

Now, applying the rule for while-loops we obtain

$$\{I\}\ \text{while true do } X := X+1\ \{I \wedge \neg(true)\}$$

false

Clearly $true \Rightarrow I$ since $I \equiv true$, and

$$I \wedge \neg(true) \Rightarrow true \wedge false$$
$$\Rightarrow false$$

Thus, by the consequence rule we conclude

$$\{true\}\ \text{while true do } X := X+1\ \{false\}$$

3.A From Mike Sheldon

- For each $X := a$ containing n's (not of the form $X := n$), replace each n with (new) T_i and put the comment $T_i := n$ before the command $X := a$.
- For each $X := Y$, replace with (new) T_i: $T_i := 0;\ X := Y + T_i$
- For each $X := a$ with "a"'s parse tree depth > 2 (i.e. larger than $X \text{ op } Y$), replace the bottom-most $X \text{ op } Y$ with new T_i and put $T_i := X \text{ op } Y$ before the (new) command $X := a'$. Repeat this last step until no AEXPS in $X := a$ have a parse tree depth > 2.

3.b. from ... ~~...~~ ...

We can show how to translate a command $c' \in Imp'$ into a command $c'' \in IBP''$ such that $c' \leq_{temp} c''$ by structural induction on c'.

By induction on c',

$c' \equiv$ skip : leave the same.

$c' \equiv X := Y+Z$: replace $X := Y+Z$ with

```
(X:=0; while ¬(X = Y+Z) do
    if (Y+Z ≥ 0) then
        X:=X+1
    else
        X:=X-1)
```

$c' \equiv X := Y-Z$: same as case for $X := Y+Z$ except replace "Y+Z" in Bexp's with "Y-Z"

$c' \equiv X := Y \times Z$; same as case for $X := Y+Z$ except replace "Y×Z" in Bexp's with "Y×Z"

$c' \equiv X := n$: replace $X := n$ with

```
(X:=0; while ¬(X=n) do
    if (n ≥ 0) then
        X:=X+1
    else
        X:=X-1)
```

$c' \equiv c_0; c_1$ or if b then c_0 else c_1 or while b do c leave the same such that c_0, c_1, and c hold from the induction hypothesis.

Problem Set 7 Solutions

Problem 1. Let $loc(B)$ be the set of locations which occur in a formula $B \in$ **Assn**.

Let $\mathcal{L}$ be a subset of **Loc**. We define an equivalence relation on states, $\sim_{\mathcal{L}}$, as follows:

$$\sigma_1 \sim_{\mathcal{L}} \sigma_2 \text{ iff } \forall X \in \mathcal{L}. \sigma_1(X) = \sigma_2(X).$$

In other words, $\sigma_1 \sim_{\mathcal{L}} \sigma_2$ iff the states σ_1 and σ_2 agree on all locations in $\mathcal{L}$.

1(a). Prove that an assertion depends only on locations explicitly mentioned in it, that is:

$$\text{if } \sigma_1 \sim_{loc(B)} \sigma_2 \text{ then } (\sigma_1 \models^I B \text{ iff } \sigma_2 \models^I B).$$

Hint: Use structural induction on B.

Solution. We first prove the following Lemma as suggested in an e-mail message to the forum.

Lemma 1. Let a be in the extended **Aexp** language (remember the cases):

$$n \mid X \mid i \mid a_0 + a_1 \mid a_0 \times a_1 \mid a_0 - a_1$$

if $\sigma_1 \sim_{\mathcal{L}} \sigma_2$, and $loc(a)$ is a subset of $\mathcal{L}$, then:

$$\mathcal{A}v[\![a]\!]I\sigma_1 = \mathcal{A}v[\![a]\!]I\sigma_2.$$

We prove this by induction on the structure of a.

1. The cases of $a \equiv n, i$ are trivial, as $\mathcal{A}v[\![a]\!]I\sigma$ does not depend on σ.
2. The case of $a \equiv X$, is interesting. Since $X \in \mathcal{L}$, $\sigma_1(X) = \sigma_2(X)$, and so by the definition of $\mathcal{A}v[\![X]\!]$, $\mathcal{A}v[\![X]\!]I\sigma_1 = \mathcal{A}v[\![X]\!]I\sigma_2$.

3. The inductive cases are easy applictions of the induction hypothesis. We do the case of $a \equiv a_1 + a_2$. Since $loc(a_1) \subseteq loc(a)$, $\sigma_1 \sim_{loc(a_1)} \sigma_2$. Thus we may apply the induction hypothesis to a_1, to get $\mathcal{A}v[\![a_1]\!]I\sigma_1 = \mathcal{A}v[\![a_1]\!]I\sigma_2$, a similar argument will get us $\mathcal{A}v[\![a_2]\!]I\sigma_1 = \mathcal{A}v[\![a_2]\!]I\sigma_2$. This lets us say:

$$\mathcal{A}v[\![a_1]\!]I\sigma_1 + \mathcal{A}v[\![a_2]\!]I\sigma_1 = \mathcal{A}v[\![a_2]\!]I\sigma_1 + \mathcal{A}v[\![a_2]\!]I\sigma_2.$$

Finally be the definition of $\mathcal{A}v[\![a_1 + a_2]\!]$, we get:

$$\mathcal{A}v[\![a_1 + a_2]\!]I\sigma_1 = \mathcal{A}v[\![a_1 + a_2]\!]I\sigma_2$$

We now prove the main part of the subproblem by an induction on the structure of B. Due to the substantial difference in **Assn** from other syntax we have used, this merits a full description of the detail this time around.

So, we do cases on the structure of B:

[$B \equiv$ **true**] Both, $\sigma_1 \models^I$ **true**, and $\sigma_2 \models^I$ **true** by definition, so $\sigma_1 \models^I B$ iff $\sigma_2 \models^I B$.

[$B \equiv$ **false**] Similar to preceeding case.

[$B \equiv a_0 = a_1$] By the Lemma, $\mathcal{A}v[\![a_0]\!]I\sigma_1 = \mathcal{A}v[\![a_0]\!]I\sigma_2 = n_0$ and $\mathcal{A}v[\![a_1]\!]I\sigma_1 = \mathcal{A}v[\![a_1]\!]I\sigma_2 = n_1$. Either $n_0 = n_1$, in which case both $\sigma_1 \models^I B$ and $\sigma_2 \models^I B$, or both $\sigma_1 \not\models^I B$ and $\sigma_2 \not\models^I B$, exactly as required.

[$B \equiv a_0 \leq a_1$] Similar to preceeding case.

[$B \equiv B_0 \wedge B_1$] $\sigma_1 \models^I B_0 \wedge B_1$ iff

$$\sigma_1 \models^I B_0 \text{ and } \sigma_1 \models^I B_1.(*)$$

Since $loc(B_0) \subseteq loc(B)$, $\sigma_1 \sim_{loc(B_0)} \sigma_2$ (similarly $\sigma_1 \sim_{loc(B_1)} \sigma_2$); by induction, $(*)$ holds iff

$$\sigma_2 \models^I B_0 \text{ and } \sigma_2 \models^I B_1.(**)$$

By the definition of $\models$, $(**)$ holds iff,

$$\sigma_2 \models^I B_0 \wedge B_1$$

Following the chain of "iff"'s, we have the required result.

[$B \equiv B_0 \vee B_1$] Similar to preceeding case.

[$B \equiv B_0 \Rightarrow B_1$] Similar to preceeding case.

[$B \equiv \neg B'$] This is not too different from the other propositional operators, but I suppose it is different enough. By definition of $\models$, $\sigma_1 \models^I \neg B'$ iff

$$\sigma_1 \not\models^I B', (*)$$

As before, $\sigma_1 \sim_{loc(B')} \sigma_2$. By induction $(*)$ holds iff

$$\sigma_2 \not\models^I B', (**)$$

Finally, by the definition of $\models$, $(**)$ holds iff $\sigma_2 \models^I \neg B'$, following the chain we have $\sigma_1 \models^I B$ iff $\sigma_2 \models^I B$.

[$B \equiv \forall i.B'$] Well, by definition of $\models$,

$$\sigma_1 \models^I \forall i.B' \text{ iff } \sigma_1 \models^{I[n/i]} B \text{ for all } n \in Int.$$

Again, $\sigma_1 \sim_{loc(B')} \sigma_2$, however this time when we use induction, we use $I[n/i]$ as the interpretation. So by induction:

$$\sigma_1 \models^{I[n/i]} B' \text{ iff } \sigma_2 \models^{I[n/i]} B'$$

to summarize:

$$\begin{aligned} \sigma_1 \models^I \forall i.B' \quad & \text{iff} \quad \sigma_1 \models^{I[n/i]} B' \text{ for all } n \in Int. \\ & \text{iff} \quad \sigma_2 \models^{I[n/i]} B' \text{ for all } n \in Int. \\ & \text{iff} \quad \sigma_2 \models^I \forall i.B' \end{aligned}$$

[$B \equiv \exists i.B'$] Similar to preceeding case. Just write "for some $n \in Int$ in place of "for all $n \in Int$."

1(b). Using part (a), give a direct semantic proof that

$$\text{if } loc(B) \cap loc_L(c) = \emptyset, \text{ then } \models \{B\}c\{B\}.$$

(Do not appeal to soundness or completeness of the Hoare rules.)

Hint: Winskel Proposition 8, p. 45.

Solution.

Suppose $loc(B) \cap loc_L(c) = \emptyset$, and $\sigma \models^I B$. We must then show that $\mathcal{C}[\![c]\!]\sigma \models^I B$. We have two cases: $\mathcal{C}[\![c]\!]\sigma$ is undefined, in which case we are done (as $\mathcal{C}[\![c]\!]\sigma \models^I B$ trivially), or $\mathcal{C}[\![c]\!]\sigma = \sigma' \in \Sigma$. We now want to show that $\sigma \sim_{loc(B)} \sigma'$ so that we may use part (a).

To show that $\sigma \sim_{loc(B)} \sigma'$, we consider an arbitrary $X \in loc(B)$. By our premis that $loc(B)$ and $loc_L(c)$ were disjoint, we know that $X \notin loc_L(c)$. By the equivalence of the operational and denotational semantics for **IMP** we know that $\langle c, \sigma \rangle \rightarrow \sigma'$. We can now use Winskel Proposition 8, p. 45 to conclude that $\sigma(X) = \sigma'(X)$. Since X was arbitrary, $\sigma \sim_{loc(B)} \sigma'$.

In the beginning we supposed that $\sigma \models^I B$, and we just showed that $\sigma \sim_{loc(B)} \sigma'$. So by part (a) $\sigma' \models^I B$. So, in the case that $\mathcal{C}[\![c]\!]\sigma$ is defined $\mathcal{C}[\![c]\!]\sigma \models^I B$, so, $\sigma \models^I \{B\}c\{B\}$.

Problem 2. In this problem, you will give a syntactic proof of the result of the preceding problem. Specifically, we want you to show that:

$$\text{if } loc(B) \cap loc(c) = \emptyset \text{ then } \vdash_{\text{Hoare}} \{B\}c\{B\}$$

Prove this by structural induction on c. Do so directly from the definition of the Hoare rules and axioms. (Do not appeal to the Completeness Theorem or the result of Problem 1).

Solution. We have one important Lemma:

Lemma 2. If $X \notin loc(B)$, then $B \equiv B[n/X]$ (note: here we are using $\equiv$ to denote syntactic equality).

(This is proven by first proving (by structural induction) the analogous result for the extended **Aexp** language, and then by an induction on the structure of B.)

We now prove $\vdash_{\text{Hoare}} \{B\}c\{B\}$, by induction on the structure of c, taking cases on the structure of c.

[$c \equiv$ **skip**] Trivial. By the rule for **skip**, $\vdash_{\text{Hoare}} \{B\}\,\mathbf{skip}\{B\}$.

[$c \equiv X := a$] By the rule for assignments, $\vdash_{\text{Hoare}} \{B[a/X]\}X := a\{B\}$. Since $loc(B) \cap loc_L(c) = \emptyset$, $X \notin loc(B)$. By the Lemma, $B[a/X] \equiv B$, thus $\vdash_{\text{Hoare}} \{B\}X := a\{B\}$, is precisely the same statement as $\vdash_{\text{Hoare}} \{B[a/X]\}X := a\{B\}$, and so we are done.

[$c \equiv c_0; c_1$] Since $loc_L(c_0) \subseteq loc_L(c)$, $loc(B) \cup loc_L(c_0) = \emptyset$. Thus, we may use induction to say: $\vdash_{\text{Hoare}} \{B\}c_0\{B\}$. A similar argument gives us $\vdash_{\text{Hoare}} \{B\}c_1\{B\}$. Finally, we may apply the rule for sequencing, to obtain $\vdash_{\text{Hoare}} \{B\}c_0; c_1\{B\}$.

[$c \equiv$ **if** b **then** c_0 **else** c_1] Similar uses of induction will give us: $\vdash_{\mathrm{Hoare}} \{B\}c_0\{B\}$, and $\vdash_{\mathrm{Hoare}} \{B\}c_1\{B\}$. Since $B \wedge b \Rightarrow B$ (by the definition of $\wedge$), we may use the rule of consequence to obtain: $\vdash_{\mathrm{Hoare}} \{B \wedge b\}c_0\{B\}$. Similary we can get $\vdash_{\mathrm{Hoare}} \{B \wedge \neg b\}c_1\{B\}$. Finally, we may apply the rule for conditionals to obtain:

$$\vdash_{\mathrm{Hoare}} \{B\}\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1\{B\}.$$

[$c \equiv$ **while** b **do** c'] Another use of induction will give us $\vdash_{\mathrm{Hoare}} \{B\}c'\{B\}$. Since $B \wedge b \Rightarrow B$, we can use the rule of consequence to obtain

$$\vdash_{\mathrm{Hoare}} \{B \wedge b\}c'\{B\}.$$

We can then apply the rule for while-loops to obtain $\vdash_{\mathrm{Hoare}} \{B\}c\{B \wedge \neg b\}$. Finally, since $B \wedge \neg b \Rightarrow B$, we may use the rule of consequence to obtain:

$$\vdash_{\mathrm{Hoare}} \{B\}c\{B\}.$$

Problem 3. In class we gave the following definition of the **DynAssn** abbreviation for the weakest precondition under c for $d \in$ **DynAssn**:

$$w(c, D) ::= \{\mathbf{true}\}c\{D\}$$

Note we used a small "w" in this definition. This is not to be confused with $W(c, B) \in$ **Assn**, which we defined in class for $B \in$ **Assn** (although $w(c, B)$ is equivalent to $W(c, B)$).

Prove from the definition of validity for **DynAssn**:

$$\models w((c_1; c_2), D) \Rightarrow w(c_1, w(c_2, D))$$

(Of course the converse ($\Leftarrow$) also holds, but we thought that it was enough of an exercise to prove the equivalence in one direction)

Solution. So, we must show that:

$$\models \{\mathbf{true}\}c_1; c_2\{D\} \Rightarrow \{\mathbf{true}\}c_1\{\{\mathbf{true}\}c_2\{D\}\}$$

In other words, suppose $\sigma \models \{\mathbf{true}\}c_1; c_2\{D\}$, then we must show that

$$\sigma \models \{\mathbf{true}\}c_1\{\{\mathbf{true}\}c_2\{D\}\}.$$

Since $\sigma \models \mathbf{true}$, we must show that

$$\mathcal{C}[\![c_1]\!]\sigma \models \{\mathbf{true}\}c_2\{D\}.$$

We now have, two cases, either $\mathcal{C}[\![c_1]\!]\sigma$ is undefined ($\perp$), in which case we are done as $\perp \models^I Anything$, or there is a $\sigma'' \in \Sigma$ such that $\mathcal{C}[\![c_1]\!]\sigma = \sigma''$, in which case, since $\sigma'' \models^I \textbf{true}$, we must show that $\mathcal{C}[\![c_2]\!]\sigma'' \models D$. We now again have two cases: $\mathcal{C}[\![c_2]\!]\sigma''$ is undefined (in which case we are done), or $\mathcal{C}[\![c_2]\!]\sigma'' = \sigma' \in \Sigma$, and we must show $\sigma' \models^I D$.

We now use our other premis, $\sigma \models^I \{\textbf{true}\}c_1; c_2\{D\}$, to show that $\sigma' \models^I D$. Well, since $\sigma \models^I \textbf{true}$, $\mathcal{C}[\![c_1; c_2]\!]\sigma \models^I D$. But, by the definition of $\mathcal{C}[\![c_1; c_2]\!]$,

$$\mathcal{C}[\![c_1; c_2]\!]\sigma = \mathcal{C}[\![c_2]\!](\mathcal{C}[\![c_1]\!]\sigma) = \mathcal{C}[\![c_2]\!](\sigma'') = \sigma',$$

and so $\sigma' \models D$, and we're done!!

Problem 4. An important technical fact for demonstrating the expressiveness of **Assn** comes from demonstrating that you can code sequences in **Assn**. In class we assumed there was a formula $SEQ \in$ **Assn**. Winskel describes the β-function, which is one way of implementing SEQ in **Assn**.

Instead of building up SEQ from some ingenious number theory (which is what β requires), we can code up SEQ by "string manipulation." As a first step, we observe that a **Num** can be represented as a sequence of characters (via their representation in some particular base, say base 3). We expect that everyone knows how to write a positive integer as its base 3 representation (which is some particular sequence of "0"'s, "1"'s and "2"'s). We could then perform some "string" operations on the base three representations, and then finally convert the end string s to the number which has s as its base 3 representation. It is easy to code a sequence of strings as one long string by concatenating the strings in the sequence, separated say by delimiter symbols. We do not have time to go all the way up to coding SEQ, but will just work out how to define string concatenation.

We show to think of numbers as strings and how to represent concatenation of two strings. For this problem we will use the following notation: for $n \geq 0$ we write $(n)_3$ to denote the base 3 representation of n (without leading zeros). In our informal discussion, we will use $\cdot$ do denote the concatenation of strings, for example "aab"$\cdot$"bbba"="aabbbba"

For a more concrete example of what will go on in this problem:

$$\begin{aligned} (5)_3 &= \text{"12"} \\ (21)_3 &= \text{"210"} \end{aligned}$$

$$\begin{aligned} (5)_3 \cdot (21)_3 &= \text{"12210"} \\ &= (3^4 + 2 \times 3^3 + 2 \times 3^2 + 3)_3 \\ &= (156)_3 \end{aligned}$$

Concretely, in this problem we will guide you to defining a formula $F \in \textbf{Assn}$ with free variables i, j, k such that F means: "$(i)_3 \cdot (j)_3 = (k)_3$".

4(a). Define a formula $POW3(i) \in \textbf{Assn}$, which is true iff i is a power of 3.

Hint: i is a power of 3 iff any divisor of i other than 1 must itself have 3 as a divisor.

Solution. This requires merely a direct encoding, which we do in two steps first we code up: $DIVISOR(i, j)$, which is true iff i is a divisor of j.

$$DIVISOR(i, j) ::= \exists k. j = k \times i$$

The full formula is then:

$$POW3(i) ::= \forall j.(DIVISOR(j, i) \land 2 \leq j) \Rightarrow DIVISOR(3, j)$$

4(b). In our informal language, let length(i), for $i \geq 0$ be the length of the base 3 representation of i. Find a formula $LEN(i, j) \in \textbf{Assn}$, such that LEN is true iff $j = 3^{\text{length}(i)}$.

Hint: j is the largest power of 3 with a certain relation to i.

Solution. Well the hint was on the right track; however, there was a fence post problem, I guess we were actually looking for the SMALLEST power of 3 strictly greater than i.

$$LEN(i, j) ::= POW3(j) \land i < j \land \forall k.(POW3(k) \Rightarrow j \leq k)$$

4(c). Define a formula $F(i, j, k) \in \textbf{Assn}$, which is true iff $(k)_3 = (i)_3 \cdot (j)_3$.

Hint: $k = 3^{\text{length}(j)} \times i + j$.

Solution. For this part, we just need to code up the hint.

$$F(i, j, k) ::= \exists l.((k = l \times i + j) \land LEN(j, l))$$

Sample Exercises for Quiz 3

Quiz 3 Coverage: Quiz 3 will cover material from Winskel §6.1-§7.3. The material on Problem Sets 6 and 7 (except problem 3 on Problem Set 6) represent a reasonable distribution of what we expect to cover on the exam. In addition we offer the following additional representative problems (some of which were part of an early draft of the Quiz).

Problem 1. In a certain sense the ability to express weakest preconditions as assertions gives an ability to short-circuit Hoare Logic. We define a short-circuited Hoare logic derivation, $\vdash_W$, as generated by only the single axiom:

$$\{W(c, B)\}c\{B\}$$

and the consequence rule:

$$\frac{\models (A \Rightarrow A') \quad \{A'\}c\{B'\} \quad \models (B' \Rightarrow B)}{\{A\}c\{B\}}$$

Prove that $\vdash_W$ is complete, that is, show that if $\models \{A\}c\{B\}$ then $\vdash_W \{A\}c\{B\}$.

Problem 2. Show that every $D \in$ **DynAssn** is equivalent to some $A \in$ **Assn**.

Hint: Define a translation from **DynAssn** to **Assn** by induction on the structure of $D \in$ **DynAssn**. To simplify the notation it may be helpful to use the notation $\widehat{D}$ to talk about the translation of D. For example the case of $D \equiv D_0 \wedge D_1$ is:

$$\widehat{D_0 \wedge D_1} = \widehat{D_0} \wedge \widehat{D_1}$$

Problem 3. Find an appropriate invariant to use in the while-rule for proving the following partial correctness assertion:

$$\{i = Y\}\textbf{while}\,\neg(Y = 0)\,\textbf{do}\,Y := Y - 1; X := 2 \times X\{X = 2^i\}$$

Problem 4. Define a formula $LCM \in$ **Assn** with free integer variables i, j and k, which means "i is the least common multiple of j and k," that is, we require that:

$$\sigma \models^I LCM \text{ iff } I(k) \text{ is the least common multiple of } I(i) \text{ and } I(j).$$

Hint: The least common multiple of two numbers is the smallest non-negative integer divisible by both.

Quiz 3

Instructions. This is a **closed book** exam; no notes either. For your reference, there is an appendix giving the syntax and the definition the "evaluates to" relation $\rightarrow_r$ for the language $\mathbf{IMP}_r$.

Write your solutions for all five problems on this exam sheet in the spaces provided, including your name on each sheet. Ask for further blank sheets if you need them. You may assume the results of previous parts in later parts of problems, so don't let "getting stuck" on any one part keep you from proceeding to later parts.

You have 110 minutes, GOOD LUCK!

NAME

problem	*points*	*score*
1	(20)	
2	(15)	
3	(20)	
4	(25)	
5	(20)	
Total	(100)	

Problem 1 [20 points].

Problem 1(a) [5 points]. Define a formula *DIVIDES* $\in$ **Assn** with free integer variables i, j, which means "j divides i," that is, we require that:

$$\sigma \models^I \mathit{DIVIDES} \quad \text{iff} \quad I(j) \text{ divides } I(i).$$

Problem 1(b) [7 points]. Define a formula *PRIME* $\in$ **Assn** with free integer variable i, which means "i is a prime." You may assume the result of problem 1(a).

Hint: A prime is a number larger than 1, whose only divisor greater than 1 is itself.

Problem 1(c) [8 points]. There is a while-invariant of the form

$$n_1 \times i + n_2 \times Y + n_3 \times X = 0$$

appropriate for a Hoare logic proof of the partial correctness assertion:

$$\{X = 0 \wedge 2 \times Y = i\}c\{X = i\}$$

where c is the command:

while $(Y = 0)$ **do**
  $Y := Y - 1;$
  $X := X + 1;$
  $X := X + 1$

What are the values of n_1, n_2, and n_3 (Partial credit may be awarded, please show your work)?

Problem 2 [15 points].

Problem 2(a) [5 points]. State the definition of $\sigma \models^I \{A\}c\{B\}$.

A *weakest precondition* of an assertion B under a command c is any logical formula, W, such that $\sigma \models^I W$ iff

$$\text{if } \mathcal{C}[\![c]\!]\sigma \text{ is defined, then } \mathcal{C}[\![c]\!]\sigma \models^I B.$$

Note that, by definition, all weakest preconditions of B under c are equivalent logical formulas.

Problem 2(b) [10 points]. Exhibit an $A \in \mathbf{Assn}$ such that A is a weakest precondition of B under c where:

$$\begin{aligned} c &\equiv \mathbf{if}\, X \leq Y \times Y \,\mathbf{then}\, Y := Y + Y \,\mathbf{else}\,\mathbf{skip} \\ B &\equiv Y \times Y \leq Z + 1 \end{aligned}$$

Problem 3 [20 points]. Although primality is easy to express with an arithmetic first-order formula, other familiar number-theoretic functions, *e.g.*, exponentiation, are not so straightforwardly expressible as **Assn**'s. But our study of expressiveness implies that exponentiation and indeed every function which can be computed by, or even "checked" by, an **IMP** command, is expressible by **Assn**'s.

More precisely, we shall say that a binary relation, R, on numbers is called **IMP**-*checkable* iff there is an **IMP** command which halts in precisely those states σ for which $R(\sigma(X_1), \sigma(X_2))$.

Problem 3(a) [8 points]. Explain why the relation $R(n, m)$ defined by $(n = 2^m)$ is **IMP**-checkable.

Problem 3(b) [12 points]. Show that for any **IMP**-checkable relation R, there is an $A_R \in$ **Assn**, such that

$$\sigma \models^I A_R \text{ iff } R(\sigma(X_1), \sigma(X_2))$$

Hint: Expressiveness.

The next problems concern a Hoare logic for the language $\mathbf{IMP}_r$, obtained by extending **IMP** with a **resultis** construct, as in Quiz 2. Recall that $\mathbf{IMP}_r$ evaluation contrasts with **IMP** evaluation because $\mathbf{IMP}_r$ expressions have side effects and so return *both* states as well as values. The syntax and evaluation rules of $\mathbf{IMP}_r$ are repeated in Appendix A. This is sufficient to determine the denotational semantics, since:

$$\begin{array}{lll} & \langle a, \sigma\rangle \rightarrow_r \langle n, \sigma'\rangle & \text{iff} \quad \mathcal{A}[\![a]\!]\sigma = (n, \sigma') \\ \text{and} & \langle c, \sigma\rangle \rightarrow_r \sigma' & \text{iff} \quad \mathcal{C}[\![a]\!]\sigma = \sigma' \end{array}$$

and similarly for $\mathbf{Bexp}_r$'s.

As for ordinary Hoare logic, we will need to prove the expressiveness of **Assn** for $\mathbf{IMP}_r$. (We will NOT change the definition of **Assn**! It is precisely as it was for **IMP**; so there are no commands embedded within **Assn**'s.) To do this we will need a notion of weakest precondition for *expressions*, referring to both the value and the state after evaluation. We define a *weakest precondition* for a number n and assertion B, with respect to an expression $a \in \mathbf{Aexp}_r$, to be a logical formula W which means "if a successfully evaluates, then its value is n and the final state satisfies B." More formally we have $\sigma \models^I W$ iff

$$\mathcal{A}[\![a]\!]\sigma = (n, \sigma') \quad \text{implies} \quad (I(i) = n \;\&\; \sigma' \models^I B), \text{ for all } n \in \mathbf{Num}, \sigma' \in \Sigma.$$

We can define **Assn**'s $W_r(a, i, B)$ expressing weakest preconditions for $\mathbf{Aexp}_r$'s and likewise **Assn**'s $W_r(c, B)$ for commands (and similarly for $\mathbf{Bexp}_r$'s, which we omit) by structural induction simultaneously on expressions and commands.

For example, some cases in the definition of the formulas $W_r(a, i, B)$ and $W_r(c, B) \in$ **Assn** are:

$$\begin{array}{rcl} W_r(n, i, B) & ::= & (n = i) \wedge B \\ W_r(a_1 + a_2, i, B) & ::= & \exists i_1 . \exists i_2 . (i = i_1 + i_2) \wedge W_r(a_1, i_1, (W_r(a_2, i_2, B))) \\ & & \text{where } i_1 \text{ and } i_2 \text{ are "fresh."} \\ W_r(\mathbf{skip}, B) & ::= & B \end{array}$$

Problem 4 [25 points]. Supply definitions for the following cases, assuming by structural induction the existence of **Assn**'s $W_r(\ldots)$ for subexpressions and subcommands:

Problem 4(a) [5 points]. $W_r(X, i, B)$

Problem 4(b) [10 points]. $W_r(c\,\mathbf{resultis}\,a, i, B)$

Problem 4(c) [10 points]. $W_r(X := a, B)$

Hint: A straightforward version is of the form $Qi.W(a, i, B[\cdot/\cdot])$, where Q is one of $\forall$ or $\exists$, and of course the dots need to be filled in.

Problem 5 [20 points]. In $\mathbf{IMP}_r$, all of Hoare logic is essentially embodied in the assignment axiom because c and $X := c\,\mathbf{resultis}\ X$ are equivalent commands. So we content ourselves with defining a Hoare logic just for $\mathbf{IMP}_r$ assignment statements.

We observed in the previous problem that for any $\mathbf{IMP}_r$ command c and $B \in$ **Assn**, there is a formula $W_r(c, B) \in$ **Assn** which is a weakest precondition. (The present problem does ***not*** depend on the correctness of your answer to the Problem 4(c).) The provability relation, $\vdash_r$, of the logic is determined by the following two rules:

Rule for assignments:

$$\{W_r(X := a, B)\}X := a\{B\}$$

Rule of consequence:

$$\frac{\models (A \Rightarrow A') \quad \{A'\}c\{B'\} \quad \models (B' \Rightarrow B)}{\{A\}c\{B\}}$$

Show that $\vdash_r$ is complete for partial correctness assertions about assignments. In other words, show that

$$\models \{A\}X := a\{B\} \quad \text{implies} \quad \vdash_r \{A\}X := a\{B\}.$$

Hint: You may use the fact that the assertion $A \Rightarrow W_r(c, B)$ is equivalent to the partial correctness assertion $\{A\}c\{B\}$.

A $\mathbf{IMP}_r$

A.1 $\mathbf{IMP}_r$ Syntax

We use n, sometimes with subscripts as in n_0, n_1, to denote arbitrary elements of **Num**. Similarly, we assume $X, Y \in \mathbf{Loc}$, $a \in \mathbf{Aexp}_r$, $t \in \mathbf{T} = \{\mathbf{true}, \mathbf{false}\}$, $b \in \mathbf{Bexp}_r$, $c \in \mathbf{Com}$, and $\sigma \in \Sigma =$ the set of states.

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1 \mid c\,\mathbf{resultis}\,a$$

$$b ::= t \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$$

$$c ::= \mathbf{skip} \mid X := a \mid c_0; c_1 \mid \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1 \mid \mathbf{while}\, b\, \mathbf{do}\, c$$

A.2 "Evals to" Rules for $\mathbf{Aexp}_r$

$$\langle n, \sigma\rangle \rightarrow_r \langle n, \sigma\rangle \qquad (\text{num} \rightarrow_r)$$

$$\langle X, \sigma\rangle \rightarrow_r \langle \sigma(n), \sigma\rangle \qquad (\text{loc} \rightarrow_r)$$

$$\frac{\langle a_0, \sigma\rangle \rightarrow_r \langle n_0, \sigma''\rangle, \quad \langle a_1, \sigma''\rangle \rightarrow_r \langle n_1, \sigma'\rangle}{\langle a_0 + a_1, \sigma\rangle \rightarrow_r \langle n, \sigma'\rangle} \qquad (\text{plus} \rightarrow_r)$$

where n is the sum of n_0 and n_1.

Similarly, there are rules (times $\rightarrow_r$) and (minus $\rightarrow_r$).

A.3 "Evals to" Rules for $\mathbf{Bexp}_r$

$$\langle t, \sigma\rangle \rightarrow_r \langle t, \sigma\rangle \qquad (\text{bool} \rightarrow_r)$$

$$\frac{\langle a_0, \sigma\rangle \rightarrow_r \langle n_0, \sigma''\rangle, \quad \langle a_1, \sigma''\rangle \rightarrow_r \langle n_1, \sigma'\rangle}{\langle a_0 = a_1, \sigma\rangle \rightarrow_r \langle t, \sigma'\rangle} \qquad (\text{equal} \rightarrow_r)$$

where $t \equiv \mathbf{true}$ if n_0 and n_1 are equal, otherwise $t \equiv \mathbf{false}$.

$$\frac{\langle c, \sigma\rangle \rightarrow_r \sigma'', \quad \langle a, \sigma''\rangle \rightarrow_r \langle n, \sigma'\rangle}{\langle c\,\mathbf{resultis}\,a, \sigma\rangle \rightarrow_r \langle n, \sigma'\rangle} \qquad (\text{resultis} \rightarrow_r)$$

Similarly, there is a rule ($\leq\ \rightarrow_r$).

$$\frac{\langle b,\sigma\rangle \rightarrow_r \langle t,\sigma'\rangle}{\langle \neg b,\sigma\rangle \rightarrow_r \langle t',\sigma'\rangle} \qquad (\text{not} \rightarrow_r)$$

where t' is the negation of t.

$$\frac{\langle b_0,\sigma\rangle \rightarrow_r \langle t_0,\sigma''\rangle, \quad \langle b_1,\sigma''\rangle \rightarrow_r \langle t_1,\sigma'\rangle}{\langle b_0 \wedge b_1,\sigma\rangle \rightarrow_r \langle t,\sigma'\rangle} \qquad (\text{and} \rightarrow_r)$$

where t is **true** if $t_0 \equiv$ **true** and $t_1 \equiv$ **true**, and is **false** otherwise.

Similarly, there is a rule (or $\rightarrow_r$).

A.4 "Evals to" Rules for $\mathbf{Com}_r$

$$\langle \mathbf{skip},\sigma\rangle \rightarrow_r \sigma \qquad (\text{skip} \rightarrow_r)$$

$$\frac{\langle a,\sigma\rangle \rightarrow_r \langle n,\sigma'\rangle}{\langle X := a,\sigma\rangle \rightarrow_r \sigma'[n/X]} \qquad (\text{assign} \rightarrow_r)$$

$$\frac{\langle c_0,\sigma\rangle \rightarrow_r \sigma'', \quad \langle c_1,\sigma''\rangle \rightarrow_r \sigma'}{\langle (c_0;c_1),\sigma\rangle \rightarrow_r \sigma'} \qquad (\text{seq} \rightarrow_r)$$

$$\frac{\langle b,\sigma\rangle \rightarrow_r \langle \mathbf{true},\sigma''\rangle, \quad \langle c_0,\sigma''\rangle \rightarrow_r \sigma'}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1,\sigma\rangle \rightarrow_r \sigma'} \qquad (\text{if-true} \rightarrow_r)$$

$$\frac{\langle b,\sigma\rangle \rightarrow_r \langle \mathbf{false},\sigma''\rangle, \quad \langle c_1,\sigma''\rangle \rightarrow_r \sigma'}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1,\sigma\rangle \rightarrow_r \sigma'} \qquad (\text{if-false} \rightarrow_r)$$

$$\frac{\langle b,\sigma\rangle \rightarrow_r \langle \mathbf{false},\sigma'\rangle}{\langle \mathbf{while}\ b\ \mathbf{do}\ c,\sigma\rangle \rightarrow_r \sigma'} \qquad (\text{while-false} \rightarrow_r)$$

$$\frac{\langle b,\sigma\rangle \rightarrow_r \langle \mathbf{true},\sigma''\rangle, \quad \langle c,\sigma''\rangle \rightarrow_r \sigma''', \quad \langle \mathbf{while}\ b\ \mathbf{do}\ c,\sigma'''\rangle \rightarrow_r \sigma'}{\langle \mathbf{while}\ b\ \mathbf{do}\ c,\sigma\rangle \rightarrow_r \sigma'} \qquad (\text{while-true} \rightarrow_r)$$

6.044J/18.423J: Computability, Programming, and Logic — Handout 36
Massachusetts Institute of Technology — 22 November 1991

Quiz 3 Solutions

Instructions. This was a **closed book** exam; no notes either. The original exam had one appendix, giving the syntax and the definition the "evaluates to" relation $\rightarrow_r$ for the language $\mathbf{IMP}_r$.

Write your solutions for all five problems on this exam sheet in the spaces provided, including your name on each sheet. Ask for further blank sheets if you need them. You may assume the results of previous parts in later parts of problems, so don't let "getting stuck" on any one part keep you from proceeding to later parts.

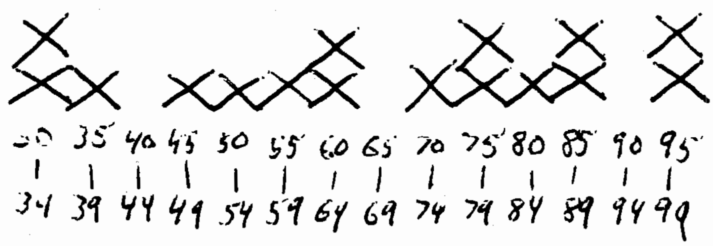
You had 110 minutes.

The exam was graded out of a possible total of 100 points. The point values are indicated on each problem. The overall statistics are as follows:

Number Submitted	16
High	99
Low	31
Median	68
Mean	67
St. Dev.	22.4

Also written on the graded exams is a projected final grade. The projections are based on the expectation that the last exam score would be roughly comparable to previous exams, and that the last problem set(s) will be turned in, with a performance comparable to the other homeworks.

The following is a histogram of the grade distribution for this Quiz:



Problem 1 [20 points].

Problem 1(a) [5 points]. Define a formula *DIVIDES* $\in$ **Assn** with free integer variables i, j, which means "j divides i," that is, we require that:

$$\sigma \models^I \mathit{DIVIDES} \quad \text{iff} \quad I(j) \text{ divides } I(i).$$

Solution: $\exists k. j \times k = i$

Problem 1(b) [7 points]. Define a formula *PRIME* $\in$ **Assn** with free integer variable i, which means "i is a prime." You may assume the result of problem 1(a).

Hint: A prime is a number larger than 1, whose only divisor greater than 1 is itself.

Solution: $2 \leq i \land \forall j.((2 \leq j \land \mathit{DIVIDES}) \Rightarrow j = i)$

Problem 1(c) [8 points]. There is a while-invariant of the form

$$n_1 \times i + n_2 \times Y + n_3 \times X = 0$$

appropriate for a Hoare logic proof of the partial correctness assertion:

$$\{X = 0 \land 2 \times Y = i\} c \{X = i\}$$

where c is the command:

while $(Y = 0)$ **do**
  $Y := Y - 1;$
  $X := X + 1;$
  $X := X + 1$

What are the values of n_1, n_2, and n_3 (Partial credit may be awarded, please show your work)?

Solution: The expected invariant is $i = 2 \times Y + X$, so $n_1 = -1$, $n_2 = 2$ and $n_3 = 1$ (or any multiples thereof).

Problem 2 [15 points].

Problem 2(a) [5 points]. State the definition of $\sigma \models^I \{A\}c\{B\}$.

Solution: if $\sigma \models^I A$, and if $\mathcal{C}[\![c]\!]\sigma$ is defined, then $\mathcal{C}[\![c]\!]\sigma \models^I B$.

A *weakest precondition* of an assertion B under a command c is any logical formula, W, such that $\sigma \models^I W$ iff

$$\text{if } \mathcal{C}[\![c]\!]\sigma \text{ is defined, then } \mathcal{C}[\![c]\!]\sigma \models^I B.$$

Note that, by definition, all weakest preconditions of B under c are equivalent logical formulas.

Problem 2(b) [10 points]. Exhibit an $A \in$ **Assn** such that A is a weakest precondition of B under c where:

$$\begin{aligned} c &\equiv \mathbf{if}\, X \le Y \times Y\, \mathbf{then}\, Y := Y + Y\, \mathbf{else}\, \mathbf{skip} \\ B &\equiv Y \times Y \le Z + 1 \end{aligned}$$

Solution: The best way to solve this problem was to chug through the definition of the formula $W(c, B)$ given in class. So:

$$\begin{aligned} W(c,B) &= (W(Y := Y + Y, B) \wedge X \le Y \times Y) \\ &= \vee(W(\mathbf{skip}, B) \wedge \neg(X \le Y \times Y)) \\ W(Y := Y + Y, B) &= B[Y + Y/Y] \quad \equiv \quad (Y + Y) \times (Y + Y) \le Z + 1 \\ W(\mathbf{skip}, B) &= B \end{aligned}$$

and so anything logically equivalent to:

$$((Y+Y) \times (Y+Y) \le Z+1 \wedge X \le Y \times Y) \vee (Y \times Y \le Z+1) \wedge \neg(X \le Y \times Y))$$

is acceptable.

Problem 3 [20 points]. Although primality is easy to express with an arithmetic first-order formula, other familiar number-theoretic functions, *e.g.*, exponentiation, are not so straightforwardly expressible as **Assn**'s. But our study of expressiveness implies that exponentiation and indeed every function which can be computed by, or even "checked" by, an **IMP** command, is expressible by **Assn**'s.

More precisely, we shall say that a binary relation, R, on numbers is called **IMP**-*checkable* iff there is an **IMP** command which halts when run on precisely those states σ for which $R(\sigma(X_1), \sigma(X_2))$.

Problem 3(a) [8 points]. Explain why the relation $R(n, m)$ defined by $(n = 2^m)$ is **IMP**-checkable.

Because the following **IMP** command c halts when run on precisely those states σ for which $R(\sigma(X_1), \sigma(X_2))$. The key to an acceptable solution is a convincing argument that such a command does exist. Clearly exhibiting one will do the job.

$X_3 := 1;$
while $1 \le X_2$ **do**
  $X_3 := X_3 \times 2;$
  $X_2 := X_2 - 1;$
if $X_1 = X_3$ **then skip else(while true do skip)**

Problem 3(b) [12 points]. Show that for any **IMP**-checkable relation R, there is an $A_R \in$ **Assn**, such that

$$\sigma \models^I A_R \text{ iff } R(\sigma(X_1), \sigma(X_2))$$

Hint: Expressiveness.

Solution:

$$A_R ::= \neg W(c_R, \mathbf{false})$$

will have the required properties, where c_R is a command which checks R.

A weakest precondition of **false** under c_R, will be true in precisely those states σ in which $R(\sigma(X_1), \sigma(X_2))$ *does not* hold, the negation of $W_r(c_R, \mathbf{false})$ is satisfied by the desired set of states.

The next problems concern a Hoare logic for the language $\mathbf{IMP}_r$, obtained by extending **IMP** with a **resultis** construct, as in Quiz 2. Recall that $\mathbf{IMP}_r$ evaluation contrasts with **IMP** evaluation because $\mathbf{IMP}_r$ expressions have side effects and so return *both* states as well as values. The syntax and evaluation rules of $\mathbf{IMP}_r$ are repeated in Appendix ??. This is sufficient to determine the denotational semantics, since:

$$\langle a, \sigma \rangle \rightarrow_r \langle n, \sigma' \rangle \quad \text{iff} \quad \mathcal{A}[\![a]\!]\sigma = (n, \sigma')$$
$$\text{and} \quad \langle c, \sigma \rangle \rightarrow_r \sigma' \quad \text{iff} \quad \mathcal{C}[\![a]\!]\sigma = \sigma'$$

and similarly for $\mathbf{Bexp}_r$'s.

As for ordinary Hoare logic, we will need to prove the expressiveness of **Assn** for $\mathbf{IMP}_r$. (We will NOT change the definition of **Assn**! It is precisely as it was for **IMP**; so there are no commands embedded within **Assn**'s.) To do this we will need a notion of weakest precondition for *expressions*, referring to both the value and the state after evaluation. We define a ***weakest precondition*** for a number n and assertion B, with respect to an expression $a \in \mathbf{Aexp}_r$, to be a logical formula W which means "if a successfully evaluates, then its value is n and the final state satisfies B." More formally we have $\sigma \models^I W$ iff

$$\mathcal{A}[\![a]\!]\sigma = (n, \sigma') \quad \text{implies} \quad (I(i) = n \;\&\; \sigma' \models^I B), \text{ for all } n \in \mathbf{Num}, \sigma' \in \Sigma.$$

We can define **Assn**'s $W_r(a, i, B)$ expressing weakest preconditions for $\mathbf{Aexp}_r$'s and likewise **Assn**'s $W_r(c, B)$ for commands (and similarly for $\mathbf{Bexp}_r$'s, which we omit) by structural induction simultaneously on expressions and commands.

For example, some cases in the definition of the formulas $W_r(a, i, B)$ and $W_r(c, B) \in$ **Assn** are:

$$\begin{aligned} W_r(n, i, B) &::= (n = i) \wedge B \\ W_r(a_1 + a_2, i, B) &::= \exists i_1.\exists i_2.(i = i_1 + i_2) \wedge W_r(a_1, i_1, (W_r(a_2, i_2, B))) \\ &\qquad \text{where } i_1 \text{ and } i_2 \text{ are "fresh."} \\ W_r(\mathbf{skip}, B) &::= B \end{aligned}$$

Problem 4 [25 points]. Supply definitions for the following cases, assuming by structural induction the existence of **Assn**'s $W_r(\ldots)$ for subexpressions and subcommands:

Problem 4(a) [5 points]. $W_r(X, i, B)$

Solution: $i = X \wedge B$

Problem 4(b) [10 points]. $W_r(c\,\mathbf{resultis}\,a, i, B)$

Solution: $W_r(c, W_r(a, i, B))$

Problem 4(c) [10 points]. $W_r(X := a, B)$

Hint: A straightforward version is of the form $Qi.W_r(a, i, B[\cdot/\cdot])$, where Q is one of $\forall$ or $\exists$, and of course the dots need to be filled in.

Solution: $\exists i.W_r(a, i, B[i/X])$, where i is "fresh."

Problem 5 [20 points]. In $\mathbf{IMP}_r$, all of Hoare logic is essentially embodied in the assignment axiom because c and $X := c\,\mathbf{resultis}\ X$ are equivalent commands. So we content ourselves with defining a Hoare logic just for $\mathbf{IMP}_r$ assignment statements.

We observed in the previous problem that for any $\mathbf{IMP}_r$ command c and $B \in \mathbf{Assn}$, there is a formula $W_r(c, B) \in \mathbf{Assn}$ which is a weakest precondition. (The present problem does *not* depend on the correctness of your answer to the Problem 4(c).) The provability relation, $\vdash_r$, of the logic is determined by the following two rules:

Rule for assignments:

$$\{W_r(X := a, B)\}X := a\{B\}$$

Rule of consequence:

$$\frac{\models (A \Rightarrow A') \quad \{A'\}c\{B'\} \quad \models (B' \Rightarrow B)}{\{A\}c\{B\}}$$

Show that $\vdash_r$ is complete for partial correctness assertions about assignments. In other words, show that

$$\models \{A\}X := a\{B\} \quad \text{implies} \quad \vdash_r \{A\}X := a\{B\}.$$

Hint: You may use the fact that the assertion $A \Rightarrow W_r(c, B)$ is equivalent to the partial correctness assertion $\{A\}c\{B\}$.

Solution: By the assignment rule:

$$\vdash_r \{W_r(X := a, B)\}X := a\{B\}$$

By assumption, $\models \{A\}X := a\{B\}$. Since $\{A\}X := a\{B\}$, and $A \Rightarrow W_r(X := a, B)$ are equivalent, it is also the case that $\models A \Rightarrow W_r(X := a, B)$. Obviously, $\models B \Rightarrow B$, so by the rule of consequence (with $c ::= X := a$, $A' ::= W_r(X := a, B)$, and $B' ::= B$):

$$\vdash_r \{A\}X := a\{B\}$$

Problem Set 8

Reading assignment. Winskel Chapter 8-9.

Due: 6 December 1991.

Instructions. Throughout this problem set we use the function $\#(x)$ to be the function which gives us the "Gödel-number" of x. Note, we are not assuming that there is some universal scheme for Gödel-numbering all things which we might wish to Gödel-number. Rather, we will use $\#$ in a variety of contexts, each of which might be Gödel-numbering different things. In any case, the intended meaning for $\#$ will either be more clearly spelled out, or will not be relevant.

Problem 1. In this problem we consider a new kind of formula, which we call **Bform** (to stand for *boolean formula*). The set of **Bform**'s is built up inductively out of a collection of boolean variables, and the boolean connectives: $\neg, \wedge, \vee$. We will let $P, P_1, P_2, Q, \ldots$ range over **Bform**'s and we let $p, p_1, p_2, q, \ldots$ range over boolean variables. Since **Bform**'s don't have **Loc**'s or **IntVar**'s contained within them, states and our old notion of I's will not be relevant to the semantics of **Bform**'s. Instead, we will use *Boolean Interpretations*, J : boolean variables $\rightarrow \{\mathbf{true}, \mathbf{false}\}$.

We wish to have a collection of equations between **Bform**'s. But first, we define: $\mathcal{B}f[\![\cdot]\!]$: boolean interpretations $\rightarrow \{\mathbf{true}, \mathbf{false}\}$, we do so by a structural induction. Specifically:

$$
\begin{aligned}
\mathcal{B}f[\![p]\!]J &= J(p) \\
\mathcal{B}f[\![\neg p]\!]J &= \begin{cases} \mathbf{true} & \text{if } \mathcal{B}f[\![p]\!]J = \mathbf{false} \\ \mathbf{false} & \text{if } \mathcal{B}f[\![p]\!]J = \mathbf{true} \end{cases} \\
\mathcal{B}f[\![p_1 \wedge p_2]\!]J &= \begin{cases} \mathbf{true} & \text{if } \mathcal{B}f[\![p]\!]J = \mathbf{true} \text{ and } \mathcal{B}f[\![p]\!]J = \mathbf{true} \\ \mathbf{false} & \text{otherwise} \end{cases} \\
\mathcal{B}f[\![p_1 \vee p_2]\!]J &= \begin{cases} \mathbf{true} & \text{if } \mathcal{B}f[\![p]\!]J = \mathbf{true} \text{ or } \mathcal{B}f[\![p]\!]J = \mathbf{true} \\ \mathbf{false} & \text{otherwise} \end{cases}
\end{aligned}
$$

Our goal is to talk about equalities of the form $P_1 = P_2$, where $P_1, P_2 \in$ **Bform**. Our semantics for such equations is given by:

$$J \models P_1 = P_2 \quad \text{iff} \quad \mathcal{B}f[\![P_1]\!]J = \mathcal{B}f[\![P_2]\!]J$$

We say the equation $P_1 = P_2$ is valid, written $\models P_1 = P_2$, iff $J \models P_1 = P_2$ for all J.

We now have a semantics for equations between **Bform**'s, but we would also like to develop a logic ($\vdash$) for syntactically proving equalities between **Bform**'s. $\vdash$ will have the axiom of reflexivity, and the usual rules for equality:

$$P = P \qquad \text{(reflexivity)}$$

$$\frac{P_1 = P_2}{P_2 = P_1} \qquad \text{(symmetry)}$$

$$\frac{P_1 = P_2 \quad P_2 = P_3}{P_1 = P_3} \qquad \text{(transitivity)}$$

$$\left.\begin{array}{c} \dfrac{P_1 = P_2}{\neg P_1 = \neg P_2} \\ \dfrac{P_1 = P_2}{P_1 \,\mathbf{op}\, P = P_2 \,\mathbf{op}\, P} \\ \dfrac{P_1 = P_2}{P \,\mathbf{op}\, P_1 = P \,\mathbf{op}\, P_2} \end{array}\right\} \text{(congruence)}$$

for $\mathbf{op} \in \{\vee, \wedge\}$.

We define the substitution of the **Bform** Q in for all occurrences of the boolean variable p, in the **Bform** R (written $R[Q/p]$) by an induction on the structure of R:

$$\begin{array}{rcl} p[Q/p] & = & Q \\ p'[Q/p] & = & p' \text{ if } p' \not\equiv p \\ (\neg R)[Q/p] & = & \neg(R[Q/p]) \\ (R_1 \wedge R_2)[Q/p] & = & (R_1[Q/p]) \wedge (R_2[Q/p]) \\ (R_1 \vee R_2)[Q/p] & = & (R_1[Q/p]) \vee (R_2[Q/p]) \end{array}$$

1(a). Show that $\vdash Q_1 = Q_2$ implies $\vdash R[Q_1/p] = R[Q_2/p]$ by structural induction on R and the definition of substitution.

1(b). Every **Bform** is equal to a formula in full disjunctive normal form, *i.e.* a sum ($\vee$) of products ($\wedge$), with all products being products of the same set of variables or their negation. By a suitable ordering of variables and terms, one can define a ***canonical form*** for **Bform**'s, such that every P is equal to a unique P' in canonical form. State very clearly such a definition of canonical form for **Bform**'s.

1(c). Write down a set of axioms which, when combined with the usual axioms and rules for equality (written at the beginning of the problem), will have the property that

$$\vdash P = Q \quad \text{iff} \quad \models P = Q$$

In addition, briefly explain why your axioms have this property.

Problem 2. The goal of this problem is to prove an important result of Tarski: that truth is not expressible in **Assn**. Specifically, we will prove, through another diagonal argument, that there is no assertion T, with $\text{FV}(T) = \{i_0\}$ (FV="free variables," see p. 84, Winskel) such that, for all $n \geq 0$,

$$T[n/i_0] \text{ is true iff } A_n \text{ is valid,}$$

where A_n is the **Assn** with Gödel-number n.

There is a function $p(m,n)$ such that $A_{p(m,n)} \equiv A_n[m/i_0]$ (where $\equiv$ denotes syntactic identity), where $p(m,n)$ can be obtained be composing pairing functions on its inputs m, and n, and using some additional constants. Thus the relation "$p(n_1, n_2) = m$" is **IMP**-checkable, and is therefore expressible in **Assn**.

Assume there was such a $T \in$ **Assn**.

2(a). Let F be an **Assn** of the form:

$$\exists i_0'.\text{“}p(i_0, i_0) = i_0'\text{”} \wedge \exists i_0.i_0' = i_0 \wedge \neg T$$

Argue that F has the property that, for all $n \geq 0$

$$\models F[n/i_0] \quad \text{iff} \quad \models \neg A_n[n/i_0]$$

2(b). Conclude, by contradiction, that no such $T \in$ **Assn** exists.

Problem 3. In this problem, we will explore another set, $\mathcal{E}$ which is not **IMP**-checkable (that is "r.e." or "recursively enumerable"), and whose complement is also not r.e.

The intuition behind $\mathcal{E}$ is the Gödel-numbered version of the problem of whether two commands halt on exactly the same inputs. Formally,

$$\mathcal{E} ::= \{n \mid \forall m.\mathcal{C}[[com_{\text{Left}(n)}]](s(m)) \text{ is defined} \quad \text{iff} \quad \mathcal{C}[[com_{\text{Right}(n)}]](s(m)) \text{ is defined}\}$$

3(a). Consider the function $p(n)$, which has the property that:

$$p(n) = \text{mkpair}(\#(X_1 := n; com_n), \#(\textbf{while true do skip}))$$

It should be pretty easy to see that the function $p(n)$ is computable by an **IMP** command, in the sense that there is some $c \in \mathbf{Com}$ that has the effect setting X_1 to $p(\sigma(X_1))$ when run on state σ, and setting a bunch of temporary registers to 0.

Suppose we had such a command c. Furthermore, suppose we had a $v \in \mathbf{Com}$ which was a verifier for $\mathcal{E}$.

Justify the statement that "$c; v$ would then be a verifier for the not-self-halt set," where

$$\text{not-self-halt} ::= \{n \mid com_n \text{ does not halt on input } n\}$$

Then conclude that $\mathcal{E}$ is not r.e.

3(b). Prove that the complement of the set $\mathcal{E}$ is not r.e. That is prove that

$$\overline{\mathcal{E}} ::= \{n \mid \exists m.\mathcal{C}[[com_{\text{Left}(n)}]](s(m)) \text{ is defined} \quad \text{iff} \quad \mathcal{C}[[com_{\text{Right}(n)}]](s(m)) \text{ is undefined}\}$$

is not r.e.

Hint: Just make a small change in the argument for 3(a).

Problem 4. Given a set of $\mathcal{S}$, we informally say "$\mathcal{S}$ is decidable" to mean that the set of Gödel-numbers of elements of $\mathcal{S}$ ($GN_{\mathcal{S}}$) is **IMP**-decidable. More formally, we define $GN_{\mathcal{S}}$ to be

$$\{\#(s) \mid s \in \mathcal{S}.\}$$

of course, this can only make sense of we have a sensible way to assign Gödel-numbers to the sorts of things which might be in $\mathcal{S}$.

We now think about how to inductively define sets of **Num**'s, using rules. So suppose we have a set of rule instances, R, whose premises and conclusions are **Num**'s. Remember what such a rule instance looks like—it is of the form X/y, where X is the set of premises, and y is the conclusion. We can view an axiom instance as a special case of a rule instance; an axiom is simply a rule with no premises (so it looks like $\emptyset/y$).

There are many different ways to Gödel-number rule instances, and which we choose does not really matter, for example we could take:

$$\#(\{x_1, \ldots, x_k\}, y) = \text{mkpair}(k, \text{mkpair}(x_1, \text{mkpair}(x_2, \cdots \text{mkpair}(x_k, y) \cdots)))$$

where

$$x_1 < x_2 \cdots < x_k$$

4(a). Suppose we have such a set rules, R, which is **IMP**-decidable. Show that the set D, of R-derivations is decidable. That is, show that

$$GN_D = \{\#(d) \mid d \text{ is an } R\text{-derivation}\}$$

is **IMP**-decidable.

Hint: From the assumed **IMP** command which can check GN_R indicate how to construct an **IMP** command deciding GN_D.

4(b). Conclude that if R is **IMP**-decidable then I_R is **IMP**-checkable.

Problem Set 8 Solutions

This handout includes some of the best solutions submitted by students for Problem Set 8. These solutions are a good representation of the level of detail expected.

In addition there are some notes about Problem 1.

Problem 1 comments. The definition of **Bform** did not include the propositional constants **true** or **false**. No points were deducted, however, if solutions inculded the use of these constants. (Note that $P \wedge \neg P$ behaves exactly like **false** and $P \vee \neg P$ behaves exactly like **true**).

Many suggestions for canonical forms did not observe the difficulty that arises because p_2 and $(p_1 \wedge p_2) \vee (\neg p_1 \wedge p_2)$ are logically equivalent, and so must have the same canonical form. The hint suggested that if we are using variables p_1 and p_2 then the canonical form of p_2 should, in fact be $(p_1 \wedge p_2) \vee (\neg p_1 \wedge p_2)$. There is a way to make p_2 the canonical form, but it is much, much harder to get right.

An alternative collection of axioms to make $\vdash$ complete could be:

The distributive laws:

$$\begin{aligned} P \wedge (Q \vee R) &= (P \wedge Q) \vee (P \wedge R) \\ P \vee (Q \wedge R) &= (P \vee Q) \wedge (P \vee R) \end{aligned}$$

The associativity laws:

$$\begin{aligned} (P \wedge (Q \wedge R)) &= ((P \wedge Q) \wedge R) \\ (P \vee (Q \vee R)) &= ((P \vee Q) \vee R) \end{aligned}$$

The commutativity laws:

$$\begin{aligned} P \vee Q &= Q \vee P \\ P \wedge Q &= Q \wedge P \end{aligned}$$

Demorgan's laws:

$$\begin{aligned} \neg(P \wedge Q) &= (\neg P) \vee (\neg Q) \\ \neg(P \vee Q) &= (\neg P) \wedge (\neg Q) \end{aligned}$$

The Idempotence laws:

$$\begin{aligned} P \wedge P &= P \\ P \vee P &= P \end{aligned}$$

Behavior of "**true**" and "**false**":

$$\begin{aligned} P \vee \neg P &= Q \vee (P \vee \neg P) \\ Q &= Q \wedge (P \vee \neg P) \\ P \wedge \neg P &= Q \wedge (P \wedge \neg P) \\ Q &= Q \vee (P \wedge \neg P) \end{aligned}$$

8/10

MICHAEL G. SHEDDO.
6.044 PS 8

1a) Proof of $(\vdash Q_1 = Q_2) \Rightarrow (\vdash R[Q_1/p] = R[Q_2/p])$ by structural induction on

BASE: case $R \equiv t$, $t \in \{true, false\}$

$\vdash t[Q_1/p] = t[Q_2/p]$
$\Leftrightarrow \vdash t = t$ by ~~t[Q_1/p] = t~~ substitution
$\Leftrightarrow$ true by reflexivity rule

case $R \equiv p$

$\vdash p[Q_1/p] = t[Q_2/p]$
$\Leftrightarrow \vdash Q_1 = Q_2$ by substitution
$\Leftrightarrow$ true by assumption $\vdash Q_1 = Q_2$

case $R \equiv p'$

$\vdash p'[Q_1/p] = p'[Q_2/p]$
$\Leftrightarrow \vdash p' = p'$ by substitution
$\Leftrightarrow$ true by reflexivity

INDUCTION case $R \equiv \neg R$

$\vdash (\neg R)[Q_1/p] = (\neg R)[Q_2/p]$
$\Leftrightarrow \vdash \neg(R[Q_1/p]) = \neg(R[Q_2/p])$ by substitution
$\Leftarrow \vdash R[Q_1/p] = R[Q_2/p]$ by congruence
$\Leftrightarrow$ true by induction

case $R \equiv R_1 \text{ op } R_2$, op $\in \wedge, \vee$

$\vdash (R_1 \text{ op } R_2)[Q_1/p] = (R_1 \text{ op } R_2)[Q_2/p]$
$\Leftrightarrow \vdash R_1[Q_1/p] \text{ op } R_2[Q_1/p] = R_1[Q_2/p] \text{ op } R_2[Q_2/p]$ by substitution

so:

$$\dfrac{\dfrac{\dfrac{}{\vdash R_1[Q_1/p] = R_1[Q_2/p]}\text{induction}}{\vdash R_1[Q_1/p] \text{ op } R_2[Q_1/p] = R_1[Q_2/p] \text{ op } R_2[Q_1/p]} \quad \dfrac{\dfrac{}{\vdash R_2[Q_1/p] = R_2[Q_2/p]}\text{induction}}{\vdash R_1[Q_2/p] \text{ op } R_2[Q_1/p] = R_1[Q_2/p] \text{ op } R_2[Q_2/p]}}{\vdash R_1[Q_1/p] \text{ op } R_2[Q_1/p] = R_1[Q_2/p] \text{ op } R_2[Q_2/p]} \text{ congruence, transitivity}$$

C/I ✓

(b) If the Boolean variables range from $P_1 \dots P_N$ order the variables "alphabetically" i.e. $P_1 < \bar{P}_1 < P_2 < \bar{P}_2 \dots P_N < \bar{P}_N$. Within each product term arrange the variables alphabetically smallest first. The terms, which will be of equal length containing a variable or its complement, (not both) should be arranged in dictionary" order, i.e. ordered by the first variable followed by the second, etc. ✓

(c) $\vdash P = Q$ iff $\models P = Q$

Pf ($\Rightarrow$) because all axioms are $\models$ and rules will preserve validity

($\Leftarrow$) the following rules and axioms are sufficient so that for every

P there is a unique "canonical" form Q s.t. $\vdash P = Q$ ✓

these axioms are

commutative ✓ $A \text{ op } B = B \text{ op } A$ [op $\in \{\wedge, \vee\}$]

associative ✓ $A \text{ op} (B \text{ op } C) = (A \text{ op } B) \text{ op } C$

distributive ✓ $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$

✓ $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$

✓ $A \vee 1 = 1$ $\quad A \wedge 0 = 0$

✓ $A \vee 0 = A$ $\quad A \wedge 1 = A$

✓ $A \vee \bar{A} = 1$ $\quad A \wedge \bar{A} = 0$

✓ $A \vee A = A$ $\quad A \wedge A = A$

~~$A \vee (A \wedge B) = A$~~ $\quad A \wedge (A \vee B) = A$

$\bar{A} \vee (\bar{A} \wedge B) = \bar{A} \vee B$ $\quad \bar{A} \wedge (\bar{A} \vee B) = \bar{A} \wedge B$

$\overline{A \wedge B \wedge C} = \bar{A} \vee \bar{B} \vee \bar{C}$ $\quad \overline{A \vee B \vee C} = \bar{A} \wedge \bar{B} \wedge \bar{C}$

binary versions are enough

these rules are sufficient to rewrite any Bform P in canonical form P'. Thus if $\models P = Q$ then $P = P'$ and $Q = P'$ since canonical form is unique. Thus by reflexivity $\models P = Q \Rightarrow \vdash P = Q$.

6/10 MICHAEL G SHELDON

2a) $F[n/i_0]$

$\Leftrightarrow$ $\exists i_0'.[\text{"}p(n,n)=i_0'\text{"} \wedge \exists i_0.(i_0'=i_0 \wedge \neg T)]$ — by def. of F and substitution

$\Leftrightarrow$ $\exists i_0'.[\text{"}p(n,n)=i_0'\text{"} \wedge \neg T[i_0'/i_0]]$ — by single-point lemma and $(a=b \wedge X) \Rightarrow ((a=b) \Rightarrow X)$

$\Leftrightarrow$ $\neg T[p(n,n)/i_0]$ — by single-point lemma and $(a=b \wedge X) \Rightarrow ((a=b) \Rightarrow X)$

$\Leftrightarrow$ $\neg A_{p(n,n)}$ — by def of T (assumption of existence)

$\Leftrightarrow$ $\neg A_n[n/i_0]$ — by def of $A_{p(m,n)}$

single-point lemma, for $OP = \exists, \forall$

$$OP\, i.(i=j \Rightarrow X) \equiv X[j/i]$$

Informal proof (obvious): since bound variable i can only take on value of j on the right side of $\Rightarrow$, quantifier OP will hold iff X holds with all i's replaced by j, or $X[j/i]$

2B) LET f BE THE GÖDEL-NUMBER OF ASSN F. WE KNOW F EXISTS SINCE IT WAS DESCRIBED IN 2A). THEN FROM 2A),

$\models F[f/i_0]$ IFF $\models \neg A_f[f/i_0]$.

BUT $A_f[f/i_0] \equiv F[f/i_0]$ SINCE A_f IS GÖDEL-NUMBER OF F.

THUS WE HAVE $\models F[f/i_0]$ IFF $\models \neg F[f/i_0]$. —X—.

THIS IS A CONTRADICTION, SO NO $T \in$ ASSN CAN EXIST.

TIM? POWERS
6.044J
P.S. 8 PROBLEM 3
6 DEC 91

10/10

PROBLEM 3.

3A) ① SINCE "IT IS EASY TO SEE" THAT $P(N)$ IS COMPUTABLE BY AN IMP COMMAND, ASSUME THAT $c \in COM$ EXISTS.

② $c;v$ WOULD BE A NOT-SELF-HALT CHECKER FOR THE FOLLOWING REASONS: c SETS X_1 TO $P(\sigma(X_1))$, THAT IS, A PAIR, THE LEFT HALF OF WHICH IS COM_N RUN ON INPUT N, AND THE RIGHT HALF OF WHICH IS WHILE TRUE DO SKIP. V CHECKS IF THIS PAIR IS IN $\mathcal{E}$, THAT IS, IF COM_N RUN ON INPUT N HALTS ON THE SAME INPUTS AS WHILE TRUE DO SKIP. HOWEVER, WE KNOW THAT FOR ANY INPUT, WHILE TRUE DO SKIP DOES NOT HALT. THUS $P(N) \in \mathcal{E}$ IFF COM_N RUN ON INPUT N DOES NOT HALT. THUS $c;v$ IS A CHECKER FOR THE NOT-SELF-HALT PROBLEM.

Note that COM_N run on input N is a program which ignores its input!

③ HOWEVER, WE KNOW THAT NOT-SELF-HALT IS NOT R.E.; THAT IS, THERE CAN BE NO CHECKER FOR IT. THUS $c;v$ CAN NOT EXIST.

④ HOWEVER, FROM ①, c EXISTS. THUS v CAN NOT EXIST. SINCE THERE CAN BE NO $v \in COM$ WHICH IS A CHECKER FOR $\mathcal{E}$, $\mathcal{E}$ IS NOT CHECKABLE THUS NOT R.E.

B) DEFINE $P'(N) = MKPAIR(\#(SKIP), \#(X_1 := N; COM_N))$

① SUPPOSE THERE IS $c' \in COM$ WHICH SETS X_1 TO $P'(\sigma(X_1))$ WHEN RUN ON σ.

② SUPPOSE $v' \in COM$ CHECKS $\overline{\mathcal{E}}$.

③ THEN $c';v'$ WOULD CHECK NOT-SELF-HALT FOR THE FOLLOWING REASON: FOR ALL INPUTS, SKIP HALTS. A PAIR p IS IN $\overline{\mathcal{E}}$ ONLY IF ONE COMMAND OF THE PAIR HALTS AND ONE DOES NOT HALT FOR A GIVEN INPUT. SINCE SKIP HALTS FOR ALL INPUTS, $P'(N) \in \overline{\mathcal{E}}$ IFF COM_N RUN ON INPUT N DOES NOT HALT. THUS $c';v'$ IS A NOT-SELF-HALT CHECKER.

④ BY SIMILAR REASONING TO ③, ④ FROM PART 3A), $\overline{\mathcal{E}}$ IS NOT CHECKABLE THUS NOT R.E.

Problem 4(a)

6/10

If we have a command, gnr, that can decide if a rule is a element of a set of rules, R, then we can decide if a rule **is** an R-derivation.

For every $d \in GN_D$, there is a finite number of premises. We test the element d and its premises with the command gnr. If gnr returns false then we exit and return the value NO. Otherwise, if there are no premises we exit and return Yes. If there are premises, we test each premise with gnr and if at any point a false is returned we exit and return NO. Otherwise we continue checking all premises and their subpremises until we reach valid axioms as checked by gnr. This continues until we have no more premises to check and true has been returned by all checks of gnr. Finally, we exit and return Yes.

b) To check if a number is in the set I_R, just do the following loop:

For all $n \in Num$

    Run DECIDEGND on input n
    If output is 1
        then extract the y term
            if y = the number being checked
                then break out and output 1
                else loop
        else loop.

Of course, this loop would be written in IMP-commands.

Incompleteness and Undecidability

Let $s(n)$ be the state in which X_1 contains n and all other location s are set to zero. A set $M \subseteq \mathbf{Num}$ is **IMP**-*checkable* iff there is an **IMP** command c such that

$$n \in M \quad \text{iff} \quad \mathcal{C}[\![c]\!]s(n) \neq \perp .$$

That is, given input n in location X_1, with all other locations initially zero, command c "checks" whether n is in M and stops when its checking procedure succeeds. The command will continue checking forever (and so never succeed) if n is not in M. Checkable sets are usually referred to as "recursively enumerable" (r.e.) sets.

Closely related is the concept of an **IMP**-*decidable* set $D \subseteq \mathbf{Num}$. D is **IMP**-*decidable* iff there is an **IMP** command c such that

$$n \in M \quad \text{implies} \quad \mathcal{C}[\![c]\!]s(n) = s(1),$$

and

$$n \notin M \quad \text{implies} \quad \mathcal{C}[\![c]\!]s(n) = s(0).$$

That is, given input n, command c tests whether $n \in M$, returning output 1 in location X_1 if so, and returning output 0 otherwise. It terminates with such an output for *all* inputs. Decidable sets are sometimes called "recursive" sets.

If c is a "decider" for M, then

$$c; \mathbf{if}\, X_1 = 1\, \mathbf{then\, skip\, else\, diverge}$$

is a "checker" for M, where $\mathbf{diverge} ::= \mathbf{while\, true\, do\, skip}$. Thus:

Lemma 1. If M is decidable, then M is checkable.

Exercise 1. Show that if M is decidable, so is the complement $\overline{M}$ of M. ($\overline{M} = \mathbf{Num} - M$.)

Exercise 2. Show that if M is checkable, then there is a checker c for M such that $\mathcal{C}[\![c]\!]s(n) \neq \perp$ implies $\mathcal{C}[\![c]\!]s(n) = s(0)$ for all $n \in \mathbf{Num}$. In other words, c only halts after it has "cleaned up all its locations."

Conversely, if c_1 is a checker for M, and c_2 is a checker for $\overline{M}$, then by constructing a command c which "time-shares" or "dovetails" c_1 and c_2, one gets a decider for M.

In a little more detail, here is how c might be written: Let T, F, S be "fresh" locations not in $\mathbf{Loc}(c_1) \cup \mathbf{Loc}(c_2)$. Let "Clear$_i$" abbreviate a sequence of assignments setting $\mathbf{Loc}(c_i) - \{X_1\}$ to 0. Then c might be:

```
T := X_1;                                          % save X_1 in T
F := 0;                                            % F is a flag
S := 1;                                            % how many steps to try
[while F = 0 do
    Clear_1; X_1 := T;
    "do c_1 for S steps or until c_1 halts";
    if "c_1 has halted in ≤ S steps" then
                                   F := 1;         % all done
                                   X_1 := 1;       % T is in M
                          else S := S + 1;         % increase the step counter
    if F = 1 then skip else
        Clear_2; X_1 := T;
        "do c_2 for S steps or until c_2 halts";
        if "c_2 has halted in ≤ S steps" then
                                   F := 1;         % all done
                                   X_1 := 0;       % T is not in M
                          else S := S + 1];        % increase the step counter
    Clear_1; Clear_2; T := 0; F := 0; S := 0       % clean up except for X_1
```

Exercise 3. Describe how to transform a command c_1 into one which meets the description "do c_1 for S steps or until c_1 halts (whichever happens first)."

So we have

Theorem 1. M is decidable iff M and $\overline{M}$ are checkable.

Let com_0, com_1, ..., com_n, ...be a list of all possible **IMP** commands. The details of how numbers are assigned to commands does not matter for the moment.

Exercise 4. Why is it "obvious" that there are only a countable number of **IMP** commands, even before an orderly way to assign "Gödel-numbers" to commands has been developed?

Let $H \subseteq \mathbf{Num}$ be the "self-halting" set:

$$H = \{ n \geq 0 \mid \mathcal{C}[\![com_n]\!]s(n) \neq \perp \}.$$

Theorem 2. $\overline{H}$ is not **IMP**-checkable.

Proof: Suppose c was an **IMP**-command which checked $\overline{H}$. That is, for all $n \in \mathbf{Num}$

$$(n < 0 \text{ or } \mathcal{C}[\![com_n]\!]s(n) = \perp) \quad \text{iff} \quad n \in \overline{H} \quad \text{iff} \quad \mathcal{C}[\![c]\!]s(n) \neq \perp .$$

Now c must appear somewhere in the list of commands, say as com_{743}. We have for all $n \geq 0$,

$$\mathcal{C}[\![com_n]\!]s(n) = \perp \quad \text{iff} \quad \mathcal{C}[\![com_{743}]\!]s(n) \neq \perp .$$

Now let $n = 743$ and we have reached a contradiction. ■

Corollary 1. Neither H nor $\overline{H}$ is **IMP**-decidable.

Notice that H depends on the order in which commands are listed, e.g., they can be listed with numerous repetitions, as long as every command appears at least once. *Any* $\overline{H}$ obtained by picking such a listing is not **IMP**-checkable.

Exercise 5. Prove that there are an uncountable number of different sets $\overline{H}$ obtainable by varying the order in which commands are listed. (Hint: Don't make the false assumption that different listings *necessarily* yield different $\overline{H}$'s.)

Now if we use a sensible assignment of numbers to commands, it will turn out that H *is* **IMP**-checkable.

Let mkpair be a pairing function for pairs of integers. For example,

$$\text{mkpair}(n, m) = 2^{\text{sg}(n)} \cdot 3^{|n|} \cdot 5^{\text{sg}(m)} \cdot 7^{|m|}$$

will serve, where

$$\text{sg}(n) = \begin{cases} 1 & \text{if } n \geq 0, \\ 0 & \text{if } n < 0. \end{cases}$$

The details of the pairing function don't matter; the important point is that there are functions "left" and "right" such that

$$\begin{aligned} \text{left}\,(\text{mkpair}(n, m)) &= n, \\ \text{right}\,(\text{mkpair}(n, m)) &= m, \end{aligned}$$

and *moreover* there are **IMP** commands which act like assignment statements of each of the forms

$$\begin{aligned} X &:= \text{mkpair}(Y, Z), \\ X &:= \text{left}(Y), \text{ and} \\ X &:= \text{right}(Y). \end{aligned}$$

Exercise 6. Let c be a text which is of the form of an **IMP** command, except that c contains assignment statements of the form "$X :=$ left(Y)." Describe how to construct an authentic **IMP** command $\hat{c}$ which simulates c up to temporary locations (cf. Problem Set 6, Problem 3, Handout 27); notation $\hat{c} \preceq_{\mathsf{temp}} c$.

Exercise 7. Suppose that the definition of **Aexp**, and hence of **IMP**, was modified to allow **Aexp**'s of the form "mkpair(a_1, a_2)," "left(a)" and "right(a)" for a, a_1, a_2 themselves modified **Aexp**'s. Call the resulting language **IMP**$'$. Explain how to translate every $c' \in \mathbf{Com}'$ into a $c \in \mathbf{Com}$ such that $c \preceq_{\mathsf{temp}} c'$.

To number commands, we begin by numbering **Loc**, which we assume consists of $X_1, X_2, \ldots$. We use 0 as the "location-tag" and define

$$\#(X_i) = \text{mkloc}(i) = \text{mkpair}(0, i).$$

We also number numerals by using tag 1:

$$\#(n) = \text{mknum}(n) = \text{mkpair}(1, n).$$

We proceed to number **Aexp**'s by using 2, 3, 4 as tags for sums, differences, and products, for example:

$$\#(a_1 + a_2) = \text{mksum}(\#a_1, \#a_2) = \text{mkpair}\,(2, \text{mkpair}(\#a_1, \#a_2))\,.$$

We number **Bexp**'s using tags 5, 6, 7, 8, 9 for $\leq$, $=$, $\wedge$, $\vee$, $\neg$, for example:

$$\#(a_1 \leq a_2) = \text{mkleq}(\#a_1, \#a_2) = \text{mkpair}\,(5, \text{mkpair}(\#a_1, \#a_2))\,,$$

$$\#(b_1 \vee b_2) = \text{mkor}(\#b_1, \#b_2) = \text{mkpair}\,(8, \text{mkpair}(\#b_1, \#b_2))\,.$$

Finally, number **Com** using tags 10–14 for $:=$, **skip**, **if**, **sequencing**, **while**, e.g.,

$$\begin{aligned} \#(\textbf{if}\, b\, \textbf{then}\, c_0\, \textbf{else}\, c_1) &= \text{mkif}(\#b, \#c_0, \#c_1) \\ &= \text{mkpair}\,(12, \text{mkpair}\,(\#b, \text{mkpair}(\#c_0, \#c_1)))\,. \end{aligned}$$

We now define a specific listing of commands using this numbering:

$$com_n = \begin{cases} c & \text{if } \#c = n, \\ \textbf{skip} & \text{otherwise.} \end{cases}$$

This definition is ok because $\#c$ uniquely determinines c. This method of numbering syntactic or finitely structured objects was first used by the great logician Kurt Gödel in the 1930's. $\#c$ is called the *Gödel number* of c.

Now that commands are numbered, it makes sense to talk about supplying a command as an "input" to another command, namely, supply its number. It is nowadays a commonplace idea (although it was a strikingly imaginative one in the 1930's) that one can write a "simulator" for **IMP** commands; in fact, the simulator itself could be programmed in **IMP**. That is, we want a command **SIM** which, given input mkpair(n_1, n_2), will give the same output as com_{n_1} running on input n_2. Using X_1 for input and output, the precise specification is

$$\mathcal{C}[\![\mathbf{SIM}]\!]s(\text{mkpair}(n_1, n_2)) = \begin{cases} \bot & \text{if } \mathcal{C}[\![com_{n_1}]\!]s(n_2) = \bot, \\ s(k) & \text{otherwise,} \end{cases}$$

where

$$k = (\mathcal{C}[\![com_{n_1}]\!]s(n_2))\,(X_1).$$

Theorem 3 (Universal Machine Theorem). There is an **IMP** command, **SIM**, meeting the above specification for all $n_1, n_2 \in \mathbf{Num}$.

Proof: A long programming exercise to construct **SIM**, and a longer, challenging exercise to prove it works correctly. ■

Corollary 2. The self-halting set H based on the Gödel-numbering-list of commands is **IMP**-checkable.

Proof: "$X_1 :=$ mkpair(X_1, X_1); **SIM**" describes an **IMP**-checker for H. ■

A set $M \subseteq \mathbf{Num}$ is *expressible* iff there is an $A \in \mathbf{Assn}$ with no locations and only one free integer variable i such that

$$\models A[n/i] \quad \text{iff} \quad n \in M.$$

In other words, the meaning of A is "i is in M." Once i is instantiated with a number, say 7, the resulting assertion $A[7/i]$ is true or false (depending on whether $7 \in M$) independent of the state σ or interpretation I used to determine its truth value.

Theorem 4. Every **IMP**-checkable set $M \subseteq \mathbf{Num}$ is expressible.

Proof:

Let $c \in com$ be an M checker, and let $\vec{Y}$ be a list of **Loc**(c) except for X_1. Let $W(c, \mathbf{false}) \in \mathbf{Assn}$ mean the weakest precondition of **false** under c. Then A expresses M where A is:

$$(\neg W(c, \mathbf{false}))\,[\vec{0}/\vec{Y}][i/X_1].$$

■

Now suppose $A_0, A_1, A_2, \ldots,$ is a list of all the location-free *closed* **Assn**'s. Such assertions are either true or false independent of the state and interpretation. We let

$$\textbf{Truth} = \{ n \geq 0 \mid \models A_n \}.$$

Now if there were a theorem proving system which was powerful enough to prove *all* (and of course, *only*) the true **Assn**'s, then we would expect to be able to write a program which given input n, searched exhaustively for a proof of A_n, and halted iff it found such a proof. Such a program would thus be a **Truth** checker.

Put another way, any system which could reasonably be called a "theorem-prover" would provide a notion of how to *decide* if some structured finite object—usually a finite sequence of **Assn**'s—was a "proof" of a given assertion. A provability checker would work by exhaustively searching through the structured finite objects to find a proof object. Thus, in order to be worthy of the name "theorem-prover," we insist that the set

$$\textbf{Provable} = \{ n \mid \vdash A_n \}$$

must be **IMP**-checkable.

We shall shortly show, however, that **Truth** is *not* **IMP**-checkable. Therefore, for all theorem-provers, **Provable** $\neq$ **Truth**. At best, **Provable** $\subsetneq$ **Truth**, and so, for any theorem-prover whose provable assertions are indeed true, there must be some true assertion which is not provable. So the theorem-prover cannot *completely* prove the true assertions. This is known as *Gödel's* (first) *Incompleteness Theorem*. In abstract form, it is simply:

Theorem 5. **Truth** is not checkable.

Now before proving this, we first note that the set **Truth** depends, like the self-halting set, on the order in which things are listed. In fact, if we choose a contrived way of listing all closed **Assn**'s, we could even ensure that **Truth** was decidable (**Exercise**: contrive such a list A_0, A_1, ...).

But if we assigned Gödel numbers to **Assn** just as we did for **Com**, we could obtain a list of closed, location-free assertions by letting

$$A_n = \begin{cases} A & \text{if } \#A = n \text{ and } A \text{ is closed and location-free,} \\ \textbf{true} & \text{otherwise.} \end{cases}$$

This numbering has the following important property: for any assertion A with no locations and a single free integer variable i, let $f(n) = \#(A[n/i])$; then we claim there is an **IMP** command which acts like an assignment $X := f(Y)$.

One way to see this is to assume that A is of the form

$$\exists j.\, j = i \wedge A'$$

where A' has no free occurrences of i. There is essentially no loss of generality in this assumption, since any $A \in$ **Assn** is equivalent to an assertion of the form above. Now we see that

$$f(n) = \text{mkexistential}(\#(j), \text{mkand}(\text{mkeq}(\#(j), \text{mknum}(n)), \#(A'))),$$

so $f(n)$ is definable by an **Aexp** extended with a "mkpair" operator, and therefore by an exercise above we know there is an **IMP** comand for $X := f(Y)$.

This property is the only fact about the numbering of closed assertions which we need to use to prove the Incompleteness Theorem, as we now show.

Proof of the Incompleteness Theorem:

Suppose $c \in$ **Com** was a **Truth** checker. Since the self-halting set H is checkable, there is an assertion B expressing H. That is, for all $n \in$ **Num**,

$$n \in H \quad \text{iff} \quad \models B[n/i].$$

Letting A be $\neg B$, we have

$$n \in \overline{H} \quad \text{iff} \quad \models A[n/i] \quad \text{iff} \quad f(n) \in \mathbf{Truth}$$

where $f(n)$ is the function describing substitution into A.

But then "$X_1 = f(X_1); c$" describes an $\overline{H}$ checker, a contradiction. ■

Exercise 8. Show that $\overline{\mathbf{Truth}}$ is not checkable either.

Exercise 9. Prove or give counter-examples to the claims that decidable (checkable, expressible) sets are closed under complement (union, intersection). Note, we are asking nine questions, not three.

Theorem 6 (Zero-state halting problem). Let

$$H_0 = \{\, n \geq 0 \mid \mathcal{C}[\![com_n]\!]s(0) \neq \perp \,\}.$$

Then $\overline{H}_0$ is not checkable.

Proof: Clearly, com_n halts in state $s(n)$ iff the command $X_1 := n; com_n$ halts in state $s(0)$.

Let $g(n) = \#(X_1 := n; com_n) = \text{mkseq}(\text{mkassign}(\text{mkloc}(1), \text{mknum}(n)), n)$. So $n \in H$ iff $g(n) \in H_0$. Now if c were an $\overline{H}_0$ checker, then "$X_1 := g(X_1); c$" decribes an $\overline{H}$ checker. ■

Exercise 10. Show that H_0 is checkable.

We now examine more closely what it is about **Assn**'s which makes their truth (and falsehood) not even checkable, let alone decidable. It might seem that the source of the problem was the quantifiers "$\forall$" and "$\exists$" whose checking seems to require an infinite search in order to complete a check. However, this is a case where naive intutition is misleading. The "hard part" of **Assn**'s has more to do with the interaction between additive and multiplicative properties of numbers than with quantifiers. In particular, if we let **PlusAssn**'s be assertions which do *not* contain the symbol for multiplication and likewise **TimesAssn** be assertions which do not contain the symbols for addition or subtraction, then validity for **PlusAssn**'s and also for **TimesAssn**'s is actually decidable, and there are logical systems of a familiar kind for proving all the valid **PlusAssn**'s and likewise for **TimesAssn**'s. These facts are not at all obvious, and the long, ingenious proofs won't be given in this course.

On the other hand, when we narrow ourselves to **Assn**'s without quantifiers, that is **Bexp**'s, it turns out that validity is *still* not checkable. This is an immediate consequence of the undecidability of "Hilbert's 10^{th} Problem," which is to decide, given $a \in$ **Aexp**, whether a has an integer-vector root. More precisely, let

$$H_{10} = \{\,\#(a) \mid a \in \mathbf{Aexp} \text{ and } \sigma \models a = 0 \text{ for some } \sigma \in \Sigma\,\}.$$

Theorem 7 (Matijasevic, 1970). H_{10} is not decidable.

This is one of the great results of 20^{th} century Mathematics and Logic. Matijasevic, a Russian, building on earlier work of Americans Davis, Putnam and Robinson, learned how to "program" with polynomials over the integers and so obtained this theorem. The proof uses only elementary number theory, but would take several weeks to present in lecture.

Exercise 11. Explain why H_{10} is checkable, and so $\overline{H_0}$ is not checkable.

Matijasevic actually proved the following general result:

Theorem 8 (Polynomial Programming). Let M be an r.e. set of nonnegative integers. Then there is an $a \in$ **Aexp** such that M is the set of nonnegative integers in the range of a.

Remember that an $a \in$ **Aexp** can be thought of as describing a polynomial function on the integers. In particular, the *range* of a is $\{\,\mathcal{A}[\![a]\!]\sigma \mid \sigma \in \Sigma\,\}$.

Exercise 12. Explain why the undecidability of Hilbert's 10^{th} Problem follows directly from the Polynomial Programing Theorem.

We now can conclude that the validity problem for **Assn**'s of the simple form "$\neg(a = 0)$" is not checkable. Let

$$\textbf{ValidNonEq} = \{\, \#(\neg(a = 0)) \mid a \in \textbf{Aexp} \text{ and } \models \neg(a = 0) \,\}.$$

Corollary 3. **ValidNonEq** is not checkable.

Proof: $\#(a) \in \overline{H}_{10}$ iff $\#(\neg(a = 0)) \in$ **ValidNonEq**. So

$$X_1 := \text{mkneg}\,(\text{mkeq}\,(X_1, \text{mknum}(0)))\,; c$$

would describe an $\overline{H}_{10}$ checker if c were a **ValidNonEq** checker. ■

On the other hand, an easy, informative example which is both decidable and even nicely axiomatizable are the valid ***equations***, i.e., **Assn**'s of the form "$a_1 = a_2$."

We begin by giving the inductive definition of the "provable" equations. We write $\vdash e$ to indicate that an equation e is provable.

$$\vdash a = a \qquad \text{(reflexivity)}$$

$$\frac{\vdash a_1 = a_2}{\vdash a_2 = a_1} \qquad \text{(symmetry)}$$

$$\frac{\vdash a_1 = a_2 \quad \vdash a_2 = a_3}{\vdash a_1 = a_3} \qquad \text{(transitivity)}$$

$$\frac{\vdash a_1 = a_2}{\vdash a_1 \,\textbf{op}\, a = a_2 \,\textbf{op}\, a} \qquad \text{(right congruence)}$$

$$\frac{\vdash a_1 = a_2}{\vdash a \,\textbf{op}\, a_1 = a \,\textbf{op}\, a_2} \qquad \text{(left congruence)}$$

$$\vdash (a_1 \,\textbf{op}\, a_2) \,\textbf{op}\, a_3 = a_1 \,\textbf{op}\, (a_2 \,\textbf{op}\, a_3) \text{ where } \textbf{op} \in \{+, -, \times\} \qquad \text{(associativity)}$$

$$\vdash a_1 \,\textbf{op}'\, a_2 = a_2 \,\textbf{op}'\, a_1 \text{ where } \textbf{op}' \in \{+, x\} \qquad \text{(commutativity)}$$

$$\vdash a + 0 = a \qquad \text{(+-identity)}$$

$$\vdash a \times 1 = a \qquad \text{($\times$-identity)}$$

$$\vdash a - a = 0 \qquad \text{(additive inverse)}$$

$$\vdash a - b = a + ((-1) \times b) \qquad \text{(minus-one)}$$

$$\vdash a_1 \times (a_2 + a_3) = (a_1 \times a_2) + (a_1 \times a_3) \qquad \text{(distributivity)}$$

$$\vdash (-n) = (-1) \times n \qquad \text{(negative numeral)}$$

$$\begin{array}{l} \vdash 1 + 1 = 2 \\ \vdash 2 + 1 = 3 \\ \vdash 3 + 1 = 4 \\ \quad \vdots \end{array} \qquad \text{(numeral successor)}$$

Theorem 9. $\vdash a_1 = a_2$ iff $\models a_1 = a_2$.

Proof:

($\Rightarrow$) This direction of the "iff" is called ***soundness*** of the proof system. It follows immediately from the inductive definition of "$\vdash$," once we note the familiar facts that all the rules (including the axioms regarded as rules with no antecedents) preserve validity.

($\Leftarrow$) This direction is called ***completeness*** of the proof system. The axioms and rules were selected to be sufficient to reduce every expression a to a "canonical form" $\hat{a}$. A canonical form is either "0" or a sum-of-distinct-monomials representation, with each monomial (product of locations) having its locations occurring in increasing order of subscript, and parenthesized to the left. Moreover, each monomial has a "coefficient" of the form "n" where n is a nonzero numeral, and these monomials-with-coefficients are added in decreasing order of degree (i.e., length), in alphabetical order of the monomials for monomials of the same degree, with the sum associated to the left also. ∎

For example, let a be the **Aexp** corresponding to

$$2 - ((X_3)^2 - ((X_2)^2 ((X_3 + 2X_4)X_2) + X_3X_4(X_3)^2)) .$$

Then $\hat{a}$ would be described as

$$(X_2)^3X_3 + 3(X_2)^3X_4 - (X_3)^2 + 2.$$

We have described a and $\hat{a}$ using the usual mathematical abbreviations in which parentheses and multiplication symbols are omitted, exponents indicate repeated products, etc. The canonical form $\hat{a} \in$ **Assn** would be written formally as follows:

$$(((1 \times (((X_2 \times X_2) \times X_2) \times X_3)))) + (3 \times (((X_2 \times X_2) \times X_2) \times X_4))$$
$$+ ((-1) \times (X_3 \times X_3)) + (2 \times 1).$$

Note that we regard "1" as a monomial of degree zero.

The idea is that first differences can be eliminated using the (minus-one) axiom. Then distributivity can be applied repeatedly to remove occurrences of products over sums. The result is an expression consisting of sums of products of locations and numbers. The products can be internally sorted using associativity and commutativity. Identical monomials with different coefficients can then be combined by distributivity, and a sum of numerical coefficients can be simplified to a single number using the numerical and identity axioms with associativity and commutativity. Enough said; we thus have:

Lemma 2. For every $a \in$ **Aexp**, there is a canonical form $\hat{a} \in$ **Aexp** such that $\vdash a = \hat{a}$.

We now state the following fact about polynomial functions on the integers.

Fact 1. If $\hat{a}_1$ and $\hat{a}_2$ are syntactically distinct canonical forms, then $\mathcal{A}[\![\hat{a}_1]\!] \neq \mathcal{A}[\![\hat{a}_2]\!]$.

Exercise 13 (optional). Prove the Fact.

Proof: [Completeness] We now can prove completeness. Suppose $\models a_1 = a_2$, i.e., $\mathcal{A}[\![a_1]\!] = \mathcal{A}[\![a_2]\!]$. By the Lemma, $\vdash a_i = \hat{a}_i$, so by soundness, $\models a_i = \hat{a}_i$ for $i = 1, 2$. So $\mathcal{A}[\![\hat{a}_1]\!] = \mathcal{A}[\![a_1]\!] = \mathcal{A}[\![a_2]\!] = \mathcal{A}[\![\hat{a}_2]\!]$. Then by the Fact above, $\hat{a}_1$ is actually syntactically identical to $\hat{a}_2$, so we have

$$\vdash a_1 = \hat{a}_1 \quad \text{and} \quad \vdash a_2 = \hat{a}_2$$

and by symmetry and transitivity we conclude $\vdash a_1 = a_2$. ∎

Corrections For File.

No Grades or Quizzes for 6.044J/18.423 will be available until 10AM on December 23rd. After that time, we will reply to E-mail queries addressed to `6044-staff@theory.lcs.mit.edu`. You may also contact David Jones in NE43-316 (Tech Square) on 253-5936 to find out your grades. Quiz 4 may also be picked up from David's office after this time.

Early final grades will not be available anywhere else.

Please detach and keep this sheet for your reference.

Quiz 4

Instructions. This is a **closed book** exam; no notes either.

There are four (4) problems. Write all your solutions on this exam sheet in the spaces provided, including your name on each sheet. Ask for further blank sheets if you need them. You may assume the results of previous parts in later parts of problems, so don't let "getting stuck" on any one part keep you from proceeding to later parts.

You have 120 minutes, GOOD LUCK!

NAME

problem	*points*	*score*
1	(17)	
2	(20)	
3	(28)	
4	(35)	
Total	(100)	

Problem 1 [17 points]. For any sets S, T, let $S - T$ be the set of all elements of S which are not elements of T.

Problem 1(a) [10 points]. Show that if S and T are ***decidable*** subsets of **Num**, then $S - T$ is decidable.

Problem 1(b) [7 points]. Give an example of two ***checkable*** (r.e.) subsets S and T of **Num** such that $S - T$ is not checkable. No explanation is required.

Problem 2 [20 points]. Let

$$\mathbf{Divergent} \stackrel{\text{def}}{=} \{n \geq 0 \mid [\![com_n]\!]s(k) = \perp \text{ for all } k\}.$$

Prove that **Divergent** is not checkable. (Here com_n is the command with Gödel number n, and $s(k)$ is the state with k in location X_1 and 0 in all other locations.)

Hint: $[\![com_n]\!]s(n) = \perp$ iff $[\![X_1 := n; com_n]\!]s(k) = \perp$ for all k.

Problem 3 [28 points]. An assertion A is **satisfiable** iff there exists a state σ and interpretation I such that $\sigma \models^I A$.

Problem 3(a) [10 points]. Let $\mathbf{SAT} \stackrel{\text{def}}{=} \{\#(A) \mid A \text{ is satisfiable}\}$. (Here $\#(A) \in \mathbf{Num}$ is the Gödel number of $A \in \mathbf{Assn}$. The assertion A need not necessarily be closed.) Prove that **SAT** is not checkable.

Hint: Consider closed, location-free assertions.

Let **BSAT** = $\{\#(b) \mid b \in$ **Bexp** and b is satisfiable$\}$.

Problem 3(b) [10 points]. Explain why **BSAT** *is* checkable.

Hint: We don't expect you to write an **IMP** program. Just describe in high-level terms an algorithm to ~~decide~~ Check whether or not a **Bexp** is satisfiable.

Problem 3(c) [8 points]. Prove that **BSAT** is *not* decidable.

Hint: Hilbert's 10^{th} Problem.

Problem 4 [35 points]. We consider axioms for symmetries (rigid, "in place" transformations) of an equilateral triangle. For example, given the triangle with vertices labeled as in Figure 1, we can apply

Transformation "*r*": *r*otate 120° clockwise, obtaining the triangle in Figure 2;

Transformation "*f*": *f*lip about the vertical axis, obtaining the triangle in Figure 3;

Transformation "*l*": *l*eave unchanged, obtaining the triangle in Figure 1 again.

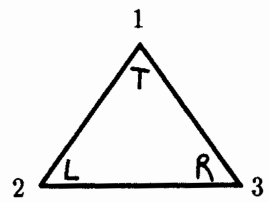


Figure 1: The original triangle.

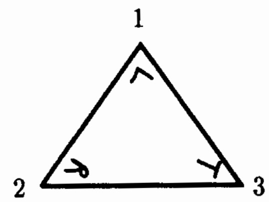


Figure 2: The original triangle after performing transformation r, a 120° clockwise rotation.

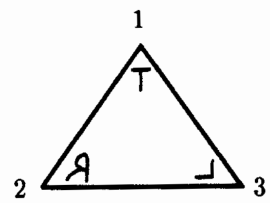


Figure 3: The original triangle after performing transformation f, a flip about the vertical axis.

Let W be the set of finite sequences (of length at least 1) of the letters r, f, and l. Elements of W are called *words* over the alphabet $\{r, f, l\}$.

By interpreting concatenation of letters as composition of permutations, we can associate with any word, w, a permutation, $[\![w]\!]$, of $\{1,2,3\}$ indicating the movement of vertices of a triangle. So the basic permutations defined by r and f are:

$$\begin{array}{llllll} [\![r]\!](1) = 2, & [\![r]\!](2) = 3, & [\![r]\!](3) = 1. \\ [\![f]\!](1) = 1, & [\![f]\!](2) = 3, & [\![f]\!](3) = 2. \end{array}$$

Note that $[\![l]\!]$ is simply the identity function. Inductively, let $[\![aw]\!] = [\![a]\!] \circ [\![w]\!]$ for $a \in \{f, r, l\}$. For example, $[\![rfrl]\!](x) = r(f(r(l(x))))$, so

$$[\![rfrl]\!](1) = 1, \quad [\![rfrl]\!](2) = 3, \quad [\![rfrl]\!](3) = 2.$$

Define "truth", $\models$, of a "triangle" word equation as follows:

$$\models (w_1 = w_2) \quad \text{iff} \quad [\![w_1]\!] = [\![w_2]\!].$$

For example, $\models rfrl = f$.

Problem 4(a) [3 points]. Exhibit w_1 and w_2, such that

$$\not\models w_1 w_2 = w_2 w_1.$$

Problem 4(b) [7 points]. The "standard" rules for equality are reflexivity, symmetry, transitivity, and congruence. State these rules for the case of word equations.

Problem 4(c) [10 points]. Show that if a sound axiom system is strong enough to prove any word equal to one of the six "canonical" forms below, then we can obtain a sound and complete axiom system by adding the standard rules for equality. The six canonical forms are:

$$l, \quad r, \quad rr, \quad f, \quad rf, \quad rrf.$$

Problem 4(d) [15 points]. Consider the proof system for triangle word equations whose rules are just the standard rules for equality plus the axioms:

$$rrr = ff = ll = l \qquad \text{(cycle)}$$

$$rl = lr = r, \quad fl = lf = f \qquad \text{(identity)}$$

$$fr = rrf \qquad \text{(swap)}$$

Briefly explain why this proof system is sound and complete. *Hint*: Show how to prove that an arbitrary word equals one of the six canonical forms of problem 4(c).

Quiz 4 Solutions

Instructions. This was a **closed book** exam; no notes either.

There are four (4) problems. Write all your solutions on this exam sheet in the spaces provided, including your name on each sheet. Ask for further blank sheets if you need them. You may assume the results of previous parts in later parts of problems, so don't let "getting stuck" on any one part keep you from proceeding to later parts.

You had 120 minutes, GOOD LUCK!

Problem 1 [17 points]. For any sets S, T, let $S - T$ be the set of all elements of S which are not elements of T.

Problem 1(a) [10 points]. Show that if S and T are ***decidable*** subsets of **Num**, then $S - T$ is decidable.

Solution A: There are two reasonable solutions to this problem. The first solution uses the fact that the set of decidable languages is closed under intersection and complement.

We observe that $S - T = S \cap \overline{T}$. In class we were told that the set of decidable languages is closed under complement, so if we can show that S and $\overline{T}$ are decidable then we are done. By the premis we have S decidable. It is then a simple task to show that if T is decidable then so is $\overline{T}$. Specifically, if d is a decider for T, then

$$d; \mathbf{if}\, X_1 = 0\, \mathbf{then}\, X_1 := 1\, \mathbf{else}\, X_1 := 0$$

is clearly an **IMP** command which decides $\overline{T}$.

Solution B: Let d_1 be a decider for S and d_2 be a decider for T, and let T_0 be a fresh location. Then the following **IMP** command is a decider for $S - T$.

$T_0 := X_1$
d_1;
if $X_1 = 0$
  then $T_0 := 0$
  else $X_1 := T_0$;
    d_2;
    $T_0 := 0$;
    if $X_1 = 0$ **then** $X_1 := 1$ **else** $X_1 := 0$

We then verbally argue that this does the job...

Problem 1(b) [7 points]. Give an example of two *checkable* (r.e.) subsets S and T of **Num** such that $S - T$ is not checkable. No explanation is required.

Solution: Let $S =$ **Num**, and T be any set which is checkable, but not decidable, for example $T = H$. Then **Num** $-T$ is simply $\overline{T} = \overline{H}$ which is not checkable.

Problem 2 [20 points]. Let

$$\textbf{Divergent} \stackrel{\text{def}}{=} \{n \geq 0 \mid [\![com_n]\!]s(k) = \perp \text{ for all } k\}.$$

Prove that **Divergent** is not checkable. (Here com_n is the command with Gödel number n, and $s(k)$ is the state with k in location X_1 and 0 in all other locations.)

Hint: $[\![com_n]\!]s(n) = \perp$ iff $[\![X := n; com_n]\!]s(k) = \perp$ for all k.

Solution: Assume $c \in$ **Com** is a checker for **Divergent**. Then the command:

$$\text{“}X_1 := \text{mkseq}(\text{mkassign}(\text{mkloc}(1), \text{mknum}(n)), n)\text{”}; c$$

will, by the hint, check NOT-SELF-HALT—a contradiction (since the NOT-SELF-SET is not checkable). Thus our assumption that **Divergent** was checkable is incorrect, and so **Divergent** is not checkable.

Problem 3 [28 points]. An assertion A is **satisfiable** iff there exists a state σ and interpretation I such that $\sigma \models^I A$.

Problem 3(a) [10 points]. Let $\textbf{SAT} \stackrel{\text{def}}{=} \{\#(A) \mid A \text{ is satisfiable}\}$. (Here $\#(A) \in$ **Num** is the Gödel number of $A \in$ **Assn**. The assertion A need not necessarily be closed.) Prove that **SAT** is not checkable.

Hint: Consider closed, location-free assertions.

The true closed, location-free assertions are not checkable. But the subset S, of Gödel-numbers of assertions which are Gödel numbers of closed, location-free assertions is a decidable set. As a closed, location-free assertion is true iff it is satisfiable then **SAT** $\cup S =$ the true closed, location-free assertions.

Suppose **SAT** were decidable. As S is obviously decidable, and decidable sets are closed under intersection, then **SAT** $\cup S$ would be decidable—which it is not. Thus **SAT** is not decidable.

Let **BSAT** = $\{\#(b) \mid b \in$ **Bexp** and b is satisfiable$\}$.

Problem 3(b) [10 points]. Explain why **BSAT** *is* checkable.

Hint: We don't expect you to write an **IMP** program. Just describe in high-level terms an algorithm to decide whether or not a **Bexp** is satisfiable.

Solution: We can evaluate B:

Just check all possible assignments of numbers to $X_1, X_2, \ldots, X_k \in loc_{(}(b))$ (there must be a finite number of locations, wlog assume these are them). If B is satisfiable, one will yeuld true and the algorithm stops. Note: it is possible to cananically odere the assignments of the locations.

When checking a particular assignment, plug the values of the X_i's into b. (This is easy to do). Then replace all **Aexp**'s in b by their value (as the **Aexp**'s no longer have locations or integer variables, this is easy to do). We can then replace all the equalities and inequalities by their appropriate truth values (again this is easy as they are of the form $n_1 \leq n_2$ or $n_1 = n_2$. Finally, we simply have a boolean combination of **true** and **false** which is also easy to evaluate. If the result is **true** then B was satisfiable, if it was **false**, we go on to the next assignment. This process of checking an assignment will always terminate, and give the right answer.

Problem 3(c) [8 points]. Prove that **BSAT** is *not* decidable.

Hint: Hilbert's 10^{th} Problem.

Solution: Suppose **BSAT** were decidable. Let d be a decider for **BSET**. As satisfiability of polynomial equalities is a special case of **BSAT**, the following command would be a decider for Hilbert's 10^{th} Problem.

$$\text{“}X_1 := \text{mkeq}(X_1, \text{mknum}(0))\text{”}; d$$

Since there can be no decider for Hilbert's 10^{th} Problem, we have a contradiction, and so **BSAT** is not decidable.

Problem 4 [35 points]. We consider axioms for symmetries (rigid, "in place" transformations) of an equilateral triangle. For example, given the triangle with vertices labeled as in Figure 1, we can apply

Transformation "r": $r$otate 120° clockwise, obtaining the triangle in Figure 2;

Transformation "f": $f$lip about the vertical axis, obtaining the triangle in Figure 3;

Transformation "l": $l$eave unchanged, obtaining the triangle in Figure 3 again.

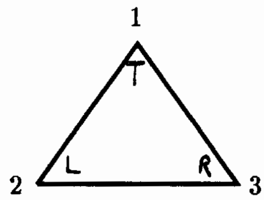


Figure 1: The original triangle.

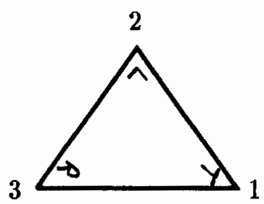


Figure 2: The original triangle after performing transformation r, a 120° clockwise rotation.

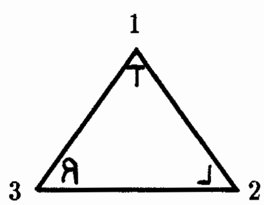


Figure 3: The original triangle after performing transformation f, a flip about the vertical axis.

Let W be the set of finite sequences (of length at least 1) of the letters r, f, and l. Elements of W are called *words* over the alphabet $\{r, f, l\}$.

By interpreting concatenation of letters as composition of permutations, we can associate with any word, w, a permutation, $[\![w]\!]$, of $\{1,2,3\}$ indicating the movement of vertices of a triangle. So the basic permutations defined by r and f are:

$$\begin{array}{lllllllll} [\![r]\!](1) & = & 2, & [\![r]\!](2) & = & 3, & [\![r]\!](3) & = & 1. \\ [\![f]\!](1) & = & 1, & [\![f]\!](2) & = & 3, & [\![f]\!](3) & = & 2. \end{array}$$

Note that $[\![l]\!]$ is simply the identity function. Inductively, let $[\![aw]\!] = [\![a]\!] \circ [\![w]\!]$ for $a \in \{f,r,l\}$. For example, $[\![rfrl]\!](x) = r(f(r(l(x))))$, so

$$[\![rfrl]\!](1) = 1, \quad [\![rfrl]\!](2) = 3, \quad [\![rfrl]\!](3) = 2.$$

Define "truth", $\models$, of a "triangle" word equation as follows:

$$\models (w_1 = w_2) \quad \text{iff} \quad [\![w_1]\!] = [\![w_2]\!].$$

For example, $\models rfrl = f$.

Problem 4(a) [3 points]. Exhibit w_1 and w_2, such that

$$\not\models w_1 w_2 = w_2 w_1.$$

Solution: For example, $w_1 = r$ and $w_2 = f$.

Problem 4(b) [7 points]. The "standard" rules for equality are reflexivity, symmetry, transitivity, and congruence. State these rules for the case of word equations.

Solution:

$$\vdash w = w \qquad \text{(reflexivity)}$$

$$\frac{\vdash w_1 = w_2}{w_2 = w_1} \qquad \text{(symmetry)}$$

$$\frac{\vdash w_1 = w_2 \quad \vdash w_2 = w_3}{\vdash w_1 = w_3} \qquad \text{(transitivity)}$$

$$\frac{\vdash w_1 = w_2}{\vdash aw_1 = aw_2} \qquad \text{(left congruence)}$$

where $a \in \{r, f, l\}$

$$\frac{\vdash w_1 = w_2}{\vdash w_1 a = w_2 a} \qquad \text{(right congruence)}$$

where $a \in \{r, f, l\}$

Problem 4(c) [10 points]. Show that if a sound axiom system is strong enough to prove any word equal to one of the six "canonical" forms below, then we can obtain a sound and complete axiom system by adding the standard rules for equality. The six canonical forms are:

$$l, \quad r, \quad rr, \quad f, \quad rf, \quad rrf.$$

Solution: Suppose $\models w_1 = w_2$, *i.e.*, $[\![w_1]\!] = [\![w_2]\!]$. By the presumption, there are canonical forms $\hat{w}_1$ and $\hat{w}_2$ such that $\vdash w_1 = \hat{w}_1$, and $\vdash w_2 = \hat{w}_2$. Since the system is sound, $\models w_i = \hat{w}_i$. $[\![\hat{w}_1]\!] = [\![w_1]\!] = [\![w_2]\!] = [\![\hat{w}_2]\!]$.

In addition, each of the six "canonical" forms have different meanings. So, we have

$$\vdash w_1 = \hat{w}_1 \quad \text{and} \quad \vdash w_2 = \hat{w}_2$$

and by symmetry and transitivity, we conclude $\vdash w_1 = w_2$.

Problem 4(d) [15 points]. Consider the complete proof system for triangle word equations whose rules are just the standard rules for equality plus the axioms:

$$rrr = ff = ll = l \qquad \text{(unit)}$$

$$rl = lr = r, \quad fl = lf = f \qquad \text{(identity)}$$

$$fr = rrf \qquad \text{(swap)}$$

Briefly explain why this proof system is sound and complete. *Hint*: Show how to prove that an arbitrary word equals one of the six canonical forms of problem 4(c).

Solution Assuming the result of Problem 4(c), it should be clear that all we need to do us show, using the above axoims and the rules for equality, that it is possible to prove that any triangle world is equal to one of the six canonical forms.

The following process will halt and reduce an arbitrary word to a canonical form.

Step 1 Erase all l's (unless $w \equiv l$, in which case we are done) This follows from the identity axioms, plus the rules for equlity.

Step 2 Move all f's to the right. This is possible from the rules for equality and the swap axiom. So now we have a word containing only r's and f's with all f's on the right.

Step 3 Replace rrr (if it occurs) by l. This is possible from the rules for equality and the unit axiom.

Step 4 Erase all l's (unless $w \equiv l$, in which case we are done)

Step 5 If there is still rrr left in w go to Step 2.

Step 6 Replace ff (if it occurs) by l. This is possible from the rules for equality and the unit axiom.

Step 7 Erase all l's (unless $w \equiv l$, in which case we are done)

Step 8 If there is still ff left in w go to Step 5.

Clearly this will halt, as we are always making the word shorter.

Clearly if it halts it will have f's to the right of r's, if there are any l's left then the result is l. If there are r's left, they all must be adjacent on the left, thus by Steps 3 to 5, there can be no more than 2 r's. If there are f's left they all must be adjacent on the right, thus by steps 6-8, there can be no more than 1 f. This paragraph now precisely characterizes the canonical forms.

		PS4	PS4	PS5	PS5	PS5	PS6	PS6	PS6	HW
Last Name	First Name	Prob3	Prob4	Prob1	Prob2	Prob3	Prob1	Prob2	Prob3	Tots
Brown	Chris	2	6	N/A	N/A	N/A	7	8	7	96
Duda	Ken	2	8	8	9	5	10	10	9	152
Fan	Mike	8	8	8	10	1	8	8	8	123
Haseltine	Mark	5	8	10	5	3	10	10	8	134
Hornik	Josh	4	5	5	3	N/A	N/A	N/A	N/A	27
Jimerson	Will	3	6	5	2	N/A	7	7	5	76
Koon	Norman	2	9	8	5	N/A	9	8	8	132
Medina	Alex	2	6	7	3	N/A	4	9	7	102
Norton	Joe	9	10	8	5	3	10	9	10	117
Powers	Mike	N/A	N/A	N/A	N/A	N/A	6	9	7	66
Purdie	Denise	N/A	N/A	7	3	0	4	10	9	70
Rauch	Pete	10	8	10	2	2	10	7	8	125
Sheldon	Mike	7	6	10	10	3	10	10	10	158
Szafranski	Jim	5	6	3	3	4	9	9	5	101
Taylor	Larry	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	38
Wood	Sasha	4	10	10	3	N/A	9	6	8	130
Yoder	Conrad	3	5	8	3	N/A	3	8	8	104
		PS4	PS4	PS5	PS5	PS5	PS6	PS6	PS6	HW
		Prob3	Prob4	Prob1	Prob2	Prob3	Prob1	Prob2	Prob3	Tots
	# Submitte	14	14	14	14	8	15	15	15	17
	High	10	10	10	10	5	10	10	10	158
	Low	2	5	3	2	0	3	6	5	27
	Median	4.00	7.00	8.00	3.00	3.00	9.00	9.00	8.00	104.00
	Mean	4.71	7.21	7.64	4.71	2.63	7.73	8.53	7.80	103.00
	St. Dev.	2.76	1.72	2.13	2.87	1.60	2.46	1.25	1.47	37.32

SIGN UP SHEET

9/11/91

(1)

TERM: FALL TERM 91

SUBJECT: 6.044J/18.423J

name	course	year	Pr{credit}	check if: listener
	6, 18, ?	1, 2, ...	0 ≤ p ≤ 1	
Amy J Weaver	18-C	4	1	
Sasha Wood	8	3	1	
Michael C. Peck	6-3	4	1	
Ken Duda	6-1	4	0.9	
Michael Fan	6-3	4	1	
MICHAEL G. SHELDON	6-3	4	1	
DAVID MOE	6-3	4	1	
Eliseo MARTINEZ	6-3	5	1	
JEM CELEBILER	6-3	3	0.9	
Pete Rauch	18C	3	1	
Denise Purdie	6-3	4	.9	
MICHAEL POWERS	6-3	4	1.0	
SHEUNG LI	6-3	4	1	
Benz Theodore	6-3	4	1	

9/11/91 (2)

SIGN UP SHEET

TERM: FALL TERM 91

SUBJECT: 6.044J/18.423J

name	course	year	Pr S check if: credit	listener
	6, 18, ?	1, 2, ...	0≤p≤1	
Conrad Yoder	18C	4	1	
David MAW	6-3	3	1	
David Han	61, 63	4	1	
Joe Norton	6 3	4	1	
SCOTT HIRAYAMA	63	4	1	
Larry Taylor	6.3	3.5	1	
Chris K. Brown	6-3a	4	1	
Norman Koon	6-3	4	1	
Mark Haseltine	6.3	4	1	
James Szafranski	6-3	4	1	
ALEJANDRO MEDINA	6-3a	4	1	

6.044 Sign Up Sheet Fri 9/13

Name	Probability you will take class 0≤x≤1
Pete Rauch	1
MICHAEL POWERS	1
Denise Purdie	1
James Szafranski	1
Norman Koon	1
Eliseo MARTINEZ	1
Mark Haseltine	1
William D. Jimerson	1
Chris K Brown	1
SHEUNG LI	1
Sasha Wood	1
Amy Weaver	1
DAVID MOE	1
SCOTT HIRAYAMA	1
YONSON SERRANO	.9
DAVID HAU	1
Larry Taylor	1
Michael Fan	1
MICHAEL G. SHELDON	1
Conrad Yoder	1
ALEJANDRO MEDINA	1

A. Meyer

9/11/91 M.I.T. CLASS LISTS, FIRST TERM 1991-92 PAGE 1

18.423JL01 TIME: MWF1 ROOM: 2-146 (42) STUDENTS: 3

ID	NAME	YEAR	CRS	ID	NAME	YEAR	CRS
208-46-7327	RAUCH, PETER C	3	18C				
335-60-2948	WEAVER, AMY J	4	18C				
293-60-9852	YODER, CONRAD G T	4	18C				

RECEIVED
ALBERT R. MEYER
SEP 16 1991
REFER TO______
FILE______

10/02/91
TERM 92/1

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CLASS LIST

SUBJECT NUMBER	SUBJECT NAME	INSTRUCTOR
6.044J	COMPUTAB LOGIC &PROGRAM	A R MEYER

ID NUMBER	COURSE	YEAR	STUDENT NAME	TOTAL-UNITS	REG-STAT
154481852	63	4	BROWN, CHRISTOPHER K		=
214193368	63	3	CELEBILER, JEM M		C
554558975	61	4	DUDA, KENNETH J		=
089508220	63	4	FAN, MICHAEL Y		=
033568209	63	4	HASELTINE, MARK E		=
326787500	61	4	HAU, DAVID T W		=
576174096	63	4	HIRAYAMA, SCOTT K A		=
428256292	63	4	JIMERSON, WILLIAM D		=
204528278	63	4	KOON, NORMAN W		=
557197827	63	4	LI, SHEUNG L		=
453571845	63	4	MARTINEZ, ELISEO		=
725986533	63	3	MAW, DAVID S Y		=
583970685	63	4	MEDINA, ALEJANDRO J		=
353522644	63	4	NORTON, JOSEPH W		=
455571578	63	4	POWERS, MICHAEL D		=
310763010	63	4	PURDIE, DENISE A		=
254356508	63	4	RHODES, BRADLEY J		C
013663391	63	4	SHELDON, MICHAEL G		=
178649306	63	4	SZAFRANSKI, JAMES P		=
045726029	63	4	TAYLOR, LARRY C		=
028640934	63	4	THEODORE, BENZNEY		=
239510312	8A	3	WOOD, SASHA K		=

REGISTRATION STATUS: = REGISTRATION OK, C CANCELLED.

TOTAL: 20

10/02/91
TERM 92/1

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CLASS LIST

NUMBER	NAME	INSTRUCTOR
18.423J	COMPUTAB LOGIC &PROGRAM	A R MEYER ✓ NE43-315

ID NUMBER	COURSE	YEAR	STUDENT NAME	TOTAL-UNITS	REG-STAT
208467327	18C	3	RAUCH, PETER C		=
335602948	18C	4	WEAVER, AMY J		=
293609852	18C	4	YODER, CONRAD G T		=

REGISTRATION STATUS: = REGISTRATION OK, C CANCELLED. TOTAL: 3

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CLASS LIST

SUBJECT NUMBER	SUBJECT NAME	INSTRUCTOR	MIT EXT
6.044J	COMPUTAB LOGIC &PROGRAM	A R MEYER	6024

ID NUMBER	COURSE	YEAR	STUDENT NAME	TOTAL-UNITS	REG-STAT	GRADE	COMMENTS
154481852	63	4	BROWN, CHRISTOPHER K		=		
214193368	63	3	CELEBILER, JEM M		C		09/11/91
554558975	61	4	DUDA, KENNETH J		=		
089508220	63	4	FAN, MICHAEL Y		=		
033568209	63	4	HASELTINE, MARK E		=		
326787500	61	4	HAU, DAVID T W		C		09/27/91
576174096	63	4	HIRAYAMA, SCOTT K A		=		
014488386	63	4	HORNIK, JOSHUA M		=		
428256292	63	4	JIMERSON, WILLIAM D		=		
204528278	63	4	KOON, NORMAN W		=		
557197827	63	4	LI, SHEUNG L		=		
453571845	63	4	MARTINEZ, ELISEO		=		
725986533	63	3	MAW, DAVID S Y		C		09/13/91
583970685	63	4	MEDINA, ALEJANDRO J		=		
353522644	63	4	NORTON, JOSEPH W		=		
455571578	63	4	POWERS, MICHAEL D		=		
310763010	63	4	PURDIE, DENISE A		=		
254356508	63	4	RHODES, BRADLEY J		C		09/11/91
013663391	63	4	SHELDON, MICHAEL G		=		
178649306	63	4	SZAFRANSKI, JAMES P		=		
045726029	63	4	TAYLOR, LARRY C		=		
028640934	63	4	THEODORE, BENZNEY		C		10/04/91
239510312	8A	3	WOOD, SASHA K		=		

REGISTRATION STATUS: = REGISTRATION OK, C CANCELLED. TOTAL: 18

10/17/91
TERM 92/1

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CLASS LIST

SUBJECT NUMBER	SUBJECT NAME	INSTRUCTOR	MIT EXT
6.044J	COMPUTAB LOGIC &PROGRAM	A R MEYER	6024

ID NUMBER	COURSE	YEAR	STUDENT NAME	TOTAL-UNITS	REG-STAT	GRADE	COMMENTS
154481852	63	4	BROWN, CHRISTOPHER K		=		
214193368	63	3	CELEBILER, JEM M		C		09/11/91
554558975	61	4	DUDA, KENNETH J		=		
089508220	63	4	FAN, MICHAEL Y		=		
033568209	63	4	HASELTINE, MARK E		=		
326787500	61	4	HAU, DAVID T W		C		09/27/91
576174096	63	4	HIRAYAMA, SCOTT K A		=		
014488386	63	4	HORNIK, JOSHUA M		=		
428256292	63	4	JIMERSON, WILLIAM D		=		
204528278	63	4	KOON, NORMAN W		=		
557197827	63	4	LI, SHEUNG L		=		
453571845	63	4	MARTINEZ, ELISEO		=		
725986533	63	3	MAW, DAVID S Y		C		09/13/91
583970685	63	4	MEDINA, ALEJANDRO J		=		
353522644	63	4	NORTON, JOSEPH W		=		
455571578	63	4	POWERS, MICHAEL D		=		
310763010	63	4	PURDIE, DENISE A		=		
254356508	63	4	RHODES, BRADLEY J		C		09/11/91
013663391	63	4	SHELDON, MICHAEL G		=		
178649306	63	4	SZAFRANSKI, JAMES P		=		
045726029	63	4	TAYLOR, LARRY C		=		
028640934	63	4	THEODORE, BENZNEY		C		10/04/91
239510312	8A	3	WOOD, SASHA K		=		

REGISTRATION STATUS: = REGISTRATION OK, C CANCELLED.

TOTAL: 18

RECEIVED
ALBERT R. MEYER

OCT 2 1 1991

REFER TO________
FILE________

10/17/91
TERM 92/1

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CLASS LIST

SUBJECT NUMBER	SUBJECT NAME	INSTRUCTOR	MIT EXT
18.423J	COMPUTAB LOGIC &PROGRAM	A R MEYER NE 43-315	6024

ID NUMBER	COURSE	YEAR	STUDENT NAME	TOTAL-UNITS	REG-STAT	GRADE	COMMENTS
208467327	18C	3	RAUCH, PETER C		=		
335602948	18C	4	WEAVER, AMY J		C		09/23/91
293609852	18C	4	YODER, CONRAD G T		=		

REGISTRATION STATUS: = REGISTRATION OK, C CANCELLED.

TOTAL: 2

RECEIVED
ALBERT R. MEYER
OCT 21 1991
REFER TO
FILE

Quiz 1 grades.

41

Chris	Brown.	10	9	12	10	41
Ken	Duda	10	15	16	25	66
Mike	Fan.	5	13	20	25	~~58~~ 63
Mark	Anseltine	10	14	19	22	65
Scott	Hirnyand	5	1	7	5	18
Josh	Hornik	5	5	9	19	38
Will	Jimerson	10	3	10	1	24
Norman	Koon	5	15	17	17	54
Sheung	Li	5	5	5	1	16
Eliseo	Martinez	5	0	1	0	6
Alex	Medina	10	5	14	6	35
David	Moe	9	11	3	14	37
Joe	Norton	7	9	5	2	23
Mike	Powers	9	5	3	14	31
Denise	Purdie	0	9	6	15	30
Pete	Rauch	10	9	19	22	60
Mike	Sheldon	10	9	12	14	45
Jim	Szafranski	10	9	17	18	54
-ary	Taylor	10	9	13	18	50
Sasha	Wood	9	14	17	25	65
Conrad	Yoder	9	11	12	9	41

21

median 41

40.8

S.094 Pisal grades:

Chris Brown	9	7	3	10
Ken Duda	10	10	10	10
Michael Fan	10		3	4
Mark Haseltine	10		2	7
David Hau	10		N/A	N/A
Scott Hirayama	8		1	1
William Jamerson	3		2	1
Norman Koo	10	10	10	10
Sherry Li	7		2	2
Elisa Martinez	5/9		1	N/A
Alejandro Medina	10		2	4
David Moe	10	10	10	10
Joe Norton	10		8	2
Mike Powers	9		2	6
Dean Purdie	10		1	3
Pete Ranich	10		3	6
Michael Sheldon	10	10	10	10
Jim Sanfrancisco	10		2	1
Larry Taylor	10		3	3
Deniz Ney Tredaee	10		1	3
Amy Weaver	10		4	6
Jasha Wood	10		2	9
Conrad Yoder	10		4	

>20 >11 >13 >22 >20 >6 >12
24 42 18
48 90 108

w/ median [illegible]

Submission	23	22	21
high	10	10	10
Low	7	1	1
median	10	2.5	4

170
18
24
7
219

Problem Set 2 grades

		out of 20
Chris Brown	5	4
Ken Duda	7	15
Michael Fan	4	5
Mark Haseltine	8	8
Scott Hirayama	3	1
William Jimerson	3	4
Norman Koon	4	15
Alex Medina	3	14
Joe Norton	6	1
Pete Rauch	6	2
Michael Shelton	7	~~12~~ 18
Jim Szafranski	4	7
Sasha Wood	8	20
Conrad Yoder	4	6
#	14	14
High	8	20
Low	3	1
median	4.5	6.5
Average	5.1	8.1

6044 P53

Chris Brown	10	10	8
Ken Dude	~~10~~ 8	10	4
[illegible] Fox	9	9	10
Mike Aiello	9	9	10
Scott Hirayama	2	5	7
Will Jackson	0	5	8
Norman Ross	10	7	10
Jimmy L.	2	4	3
Alex Meldon	8	8	10
David Mai	9	9	10
Ted Norton	9	1	6
Mike Pappe	10	7	10
Dazz Rucker	7	6	10
Pete Raven	9	10	8
Mitchell Sheldon	10	8	10
Jim Setman	9	8	6
Larry Taylor	5	7	10
Jason Wood	10	8	7
Conrad [illegible]	10	9	10

	1	2	3
E[illegible]	9	10	14
~~High~~	10	10	10
Low	0	1	3
Median	9	8	10

Last N	First Na	Quiz1	Prob0	Prob1	Prob2	Prob1	Prob2	prob1	Prob2	Prob2	Tots
Brown	Chris	41	9	3	10	5	4	10	10	8	59
Duda	Ken	66	10	10	10	7	15	8	10	4	74
Fan	Mike	63	10	3	4	4	5	9	9	10	54
Haselt	Mark	65	10	2	7	8	8	9	9	10	63
Hiraya	Scott	18	8	1	1	3	1	2	3	7	26
Hornik	Josh	38	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0
Jimers	Will	24	8	2	1	3	4	0	6	8	32
Koon	Normar	54	10	10	10	4	15	10	7	3	69
Li	Sheung	16	7	2	2	N/A	N/A	2	6	3	22
Martin	Eliseo	6	8	1	N/A	N/A	N/A	N/A	N/A	N/A	9
Medin:	Alex	35	10	2	4	3	14	9	9	10	61
Moe	David	37	10	10	10	N/A	N/A	9	9	10	58
Nortol	Joe	23	10	8	2	6	1	9	1	6	43
Power	Mike	31	9	2	6	N/A	N/A	10	7	10	44
Purdie	Denise	30	10	1	3	N/A	N/A	7	6	10	37
Rauch	Pete	60	10	3	6	6	2	9	10	8	54
Sheldc	Mike	45	10	10	10	7	18	10	8	10	83
Szafra	Jim	54	10	2	1	4	7	9	8	6	47
Taylor	Larry	50	10	3	3	N/A	N/A	5	7	8	36
Wood	Sasha	65	10	2	2	8	20	10	8	7	67
Yoder	Conrad	41	10	4	4	4	6	10	9	10	57
	# Subn	21	20	20	19	14	14	19	19	19	21
	High	66	10	10	10	8	20	10	10	10	83
	Low	6	7	1	1	3	1	0	1	3	0
	Median	41.0	10.0	2.5	4.0	4.5	6.5	9.0	8.0	8.0	54.0
	Mean	41.0	9.5	4.1	5.1	5.1	8.6	7.7	7.5	7.8	47.4
	St. Dev	17.8	0.9	3.4	3.5	1.8	6.5	3.1	2.4	2.4	21.2

Quiz 2

Name	1	2	3	4	Total
Chris Brown	23	15	2	0	40
Ken Duda	15	25	25	16	81
Michael Fan	25	15	03	08	51
Mark Haseltine	20	25	21	08	74
Josh Hornik	25	25	23	09	82
Will Jimerson	24	15	10	00	49
Norman Koon	25	25	03	06	59
Alex Medina	06	15	03	04	28
Joe Norton	25	18	23	00	66
Michael Powers	16	15	00	06	37
Denise Purdie	25	15	5	3	48
Pete Rauch	25	25	25	00	75
Michael Sheldon	25	25	25 20 af	20	95
Jim Szafranski	25	25	(10) 20	(01) af 9	(61) ~~[illegible]~~ af 71
Larry Taylor	SEE ALBERT				
Sasha Wood	25	25	~~25~~ 20	~~20~~ 17	87
Conrad Yoder	25	15	03	00	43

(Joe Norton, Michael Powers) — Both felt spent too much time on #4 & not rep. of understanding.

Problem Set 4.

Chris Brown	N/A	7	2	6
Ken Duda	7	10	2	~~9~~ 8
Michael Fan	2	8	8	~~6~~ 8
Mark Haseltine	5	7	5	~~9~~ 8
Will Jimerson	1	8	~~9~~ 5	6
Joe Norton	2	~~6~~ 8	10	9
Pete Rauch	5	~~8~~ 9	10	~~10~~ 8
Mike Sheldon	2	7	7	~~7~~ 6
Jim Szafranski	3	7	5	6
Sasha Wood	4	9	10	9
Norman Koon	N/A	7	2	9
Conrad Yoder	4	5	3	5
Total	10	12	12	12
MAX	7	10	10	~~[illegible]~~ 9
MIN	1	5	2	~~[illegible]~~ 5
~~Fr~~ Median	3.5	7.5	5	~~[illegible]~~ 8
Josh Hornick	2	6	4	5
Alex Medina	2	3	2	6

Ps5

Ken Duda	8	9	5
Michael Fan	9	10	1
Mark Angeltine	10	5	3
Joel Hornik	5	3	N/A
Will Jimerson	5	2	N/A
Norman Koon	8	5	N/A
Alex Medina	7	3	N/A
Joe Norton	8	5	3
Denise Purdue	7	3	0
Pete Rauch	10	2	2
Michael Shen	10	10	3
Jim Szufranski	3	3	4
Sean Wood	10	3	N/A
Conrad Yoder	8	3	N/A

PS6.

Chris Brown	7	8	7
Ken Duda	10	10	9
Mike Fan	8	8	8
Mark Anseltine	10	10	8
Will Jimerson	7	7	5
Norm Koon	9	8	8
Alex Medina	4	9	7
Joe Norton	10	9	~~9~~ 10 al
Mike Powers	6	9	7
Denise Purdie	4	10	9
Mike Sheldon	10	10	10
Pete Rauch	10	7	8
Jim Szymanski	9	9	5
Sasha Wood	9	6	8
Conrad Yoder	3	8	8

add Joel to PS 4   2 8 4 3

Add Alex to PS4   2 3 2 6

PS 7 grades

~~Kent~~

Ken Duda	10	10	2	10
Mike Fan	7	9	2	9
Mark Anseltine	7	8	10	10
Josh Aurnik	4	7	N/A	N/A
~~ta~~ Will Jimeson	5	8	10	2
Norman Koon	5	7	1	8
Alex Medina	9	7	6	4
Joe Norton	9	10	1	8
Mike Powers	5	6	3	10
Denise Purdie	4	6	1	8
Pete Rauch	7	10	2	10
Michael Sheldon	7	10	2	9
Jim Szafranski	7	7	2	3
Sasha Wood	5	10	7	10
Conrad Yoder ~~Sasha Wood~~	6	10	6	8

Josh Aurnik PS 6, needs to redo! ~~3 3 0~~ 5 4 6

6.044 pz
p2
11/20/91

Chris Brown	14	08	07	05	00	34
Kendeda	20	15	20	25	20	99 (100-1)
Mike Fan	20	13	20	16	20	89
Mark Anseltine	20	09	07	13	15	64
Josh Aarnik	20	15	04	20	20	79
Will Jimerson	15	03	00	16	01	35
Norman Koon	20	08	06	21	17	72
Alex Medina	20	02	03	04	02	31
Joe Norton	15	15	08	20	20	78
Mike Powers	18	15	09	14	02	58
Denise Purdie	18	05	03	08	20	54
Pete Rauch	20	15	20	24	20	99
Jim Szafranski	19	00	02	21	20	62
Mike Sheldon	20	15	18	14	20	87
Susan Wood	14	12	20	17	20	83
Conrad Yoder	18	02	08	15	05	48

enter me

6.044 P.S.8

Chris Brown	6	6		3
Ken Duda	7	9	8 or 10 ~~9~~	3
Mike Fan	9	6	10	10
Mark Anselline	7	1	6	7
Alex Medina	6	1	2	4
Joe Norton	7	6	5	6
Mike Power	8	10	10	8
Denise Purdie	8	N/A	9	4
Pete Rauch	5	4	N/A	N/A
Michael Sheldon	8	6	6	N/A
Jim Szafranski	5	5	4	8
Sasha Wood	8	6	3	2
Conrad Yoder	3	N/A	1	1

Department of Ocean Engineering
MASSACHUSETTS INSTITUTE OF TECHNOLOGY • CAMBRIDGE, MASS. 02139

J. Kim Vandiver
Chair of the MIT Faculty
MIT Room 5-222

Tel: (617) 253-4366
FAX: (617) 253-8125
e-mail: kimv@athena.mit.edu

August 29, 1991

TO: Members of the Faculty

FROM: J. Kim Vandiver, Chair of the Faculty

SUBJECT: Beginning-of-Term and End-of-Term Regulations
Creative Uses of the Final Exam Period
Policy on Evening Exams/Quizzes in Undergraduate Subjects

I am writing to emphasize several important matters that require the attention of each faculty member teaching a graduate or undergraduate subject this term:

Beginning-of-Term Expectations

1. In accordance with the Faculty rules, exercises should, in general, be held between 9 am and 5 pm Monday through Friday. For undergraduate subjects, there can be no required academic exercises between 5 pm and 7 pm Monday through Friday.

2. Instructors are asked to provide, during the first three weeks of classes, a clear and complete description of the requirements in each subject, including the due dates for required work, the schedule of examinations during the term, and the grading criteria and procedures to be used. Major assignments should be assigned early enough to allow students the opportunity to manage their time effectively throughout the term.

3. The fact that a final examination will be given in your subject must be announced to your students before the end of the third week of the term (September 27). Final examinations are held during the final examination period following each term and must be scheduled through the Office of the Registrar. (You should already have received the form for scheduling final exams and returned it to the Registrar.) The final examination scheduled in a subject may be of any length from one hour to three hours.

For those of you who have not been in the habit of using the final examination period, I encourage you to consider an alternative use. For example, you might schedule an ordinary one-hour quiz as if it were a final exam, but allow the full three-hour examination period. This removes the time pressure as a factor in the exam and frees up one lecture period during the term. Such an exam need not count more than others during the term or be comprehensive in nature. It must, however, be scheduled with the Registrar as soon as possible and announced to your class at the beginning of the term.

4. In accordance with the "Departmental Guidelines Relating to Academic Honesty," it is your responsibility early in the term to inform students of your expectations regarding permissible academic conduct. Particular attention should be given to such questions as joint work on homework assignments, and the use of prior years' materials in completing problem sets, lab reports, and other assignments.

(over)

End-of-Term Planning

Each term, a number of subjects come to my attention which have requirements that are in conflict with the Faculty rules restricting exams and work assignments in subjects at the end of the term. While usually well-intended, requirements that are in violation of the rules often impose hardships on students, given their overall loads. When violations occur, the Chair of the Faculty has the responsibility to contact the instructor to rectify the situation. It is usually difficult and awkward to resolve such situations late in the term in a way that is fair to the students and which preserves the educational value intended by the instructor.

Since effective planning early in the term can help avoid these problems, I have summarized below the Faculty's End-of-Term Regulations. They apply to both undergraduate and graduate subjects. This term, the last day of classes is Thursday, December 12; Reading Period is December 13-15 (Friday-Sunday), followed by final exams, December 16-20 (Monday-Friday).

1. In a subject with a final exam, no other examination may be given and no assignment may fall due after Friday, December 6. Of course, regular classes and reading assignments may continue during the last week of the term (through December 12), and new material presented during this period may be covered in the final exam. The scheduled time for a final exam cannot be changed once it has been officially published; inquiries about limited exceptions to this policy should be directed promptly to the Registrar.

2. In a subject with no final exam, only one of the following may be given during the last week of classes (December 9-12): either a one-hour quiz may be given during a regularly scheduled class period or one assignment (term paper, lab report, take-home exam, problem set, oral presentation, etc.) may fall due. (A quiz of one and one-half hours is allowed, but only if done within a regular class period.)

3. It is inappropriate for comprehensive examinations (exams covering most of the term's work) to be given at any time other than during the final exam period.

4. No classes, examinations, or exercises of any kind may be scheduled beyond the end of the last regularly scheduled class in a subject, except for final exams that have been scheduled through the Registrar's Office. Any formal reviews of subjects should be held during regular class periods, but the rule does not exclude the possibility of sessions after the last day of classes at which the instructing staff is available to answer questions of students who choose to attend. (The Architecture design reviews that occur during the final exam period are considered to be equivalent to final examinations and are scheduled by the Department.)

5. No assignment, of any kind, may be given which falls due after the last regularly scheduled meeting of the class for that subject. This does not prevent an instructor from giving an extension to an individual student, but an extension should not need to be given to the majority of the class.

6. Students are entitled to expect that no Faculty member will deviate from these rules except with prior permission of the CAP for undergraduate subjects and the CGSP for graduate subjects, and that any such approved exception will be announced early in the term and emphasized appropriately. Having students vote on some deviation from the rules is not an acceptable procedure.

These regulations are intended to improve the quality of education at MIT by balancing student workloads, thereby reducing end-of-term stress. If there are questions about any of these provisions, I will be happy to help resolve them.

Policy on Evening Exams/Quizzes in Undergraduate Subjects

The following policies are **applicable to undergraduate subjects only**.

1. An evening exam must be the equivalent of a quiz that could be given in a normal one-hour class period. The duration of an evening exam may not exceed two hours. An evening exam is defined as a written exercise (quiz) that is not given in a regular class period and begins after 7:00 pm.

2. During the week that an evening exam is given, a regularly scheduled class hour (lecture or recitation) shall be cancelled; or, alternatively, no homework shall be assigned for that week. It is the intent of the Faculty that evening exams be used only to ease the time pressure on students of one-hour exams given during a regular class period, and not as a means of adding to the number of class periods in a term.

3. No evening exams or review sessions are to be scheduled on Monday evening, and faculty are urged to avoid scheduling exams and review sessions on Wednesday evening. There is a need for times when evening classes and undergraduate seminars can be scheduled free from potential conflict with evening exams.

4. When possible, evening exams should be scheduled through the Registrar's Office three weeks before Registration Day so that dates can be included on students' Class Schedules for planning purposes during the Registration process. In any event, faculty must announce the schedule of any evening exams during the first week of the term.

5. Students who have a conflict between a scheduled evening exam and other scheduled academic or extracurricular activities will be provided with an exam at an alternate time.

Thank you for your help and best wishes for a successful semester.

Eta Kappa Nu Faculty Subject Evaluation Form

Please return completed form to 38-476

This form is intended to help us make our subject evaluations more complete and accurate. Please return it promptly so that we can include the information in this term's *Underground Guide to Course VI*. The first few questions are about whether the course is true to the "official" description; we appreciate your candid response. Feel free to write more than we have provided space for on this sheet.

Subject Number: 6.044J/18.423J **Date:** 11/22

Lecturer(s): Albert Meyer **TA(s):** Arthur Lent

Recitation instructor(s):

* This Form was filled out by the TA: Arthur Lent, After Consultation w/ Albert Meyer

1. What is the subject about? Is the description in the MIT Bulletin accurate? What have you emphasized and de-emphasized this term? Also comment on the balance between application and theory.

This class is about applying rigorous mathematical principles to computer science. It is about proving general properties of programming languages and in general the absolute limitations of computers. The Course Description is accurate — although a better title might have been Programming, logic, and computability. We have emphasized aspects of the class relating to program correctness. It is a pure theory class!!

2. What are the prerequisites for this subject? (They don't have to be subjects.)

A good understanding of 18.063, and a familiarity with logical notation.

3. How much time should this subject require? Give weekly class-lab-preparation hours, 3 - ___ - 7. How hard is this class? Rate the difficulty of the subject on a scale of 0 to 10 where 0=trivial, 5=average MIT, 10=very difficult. This is not asking how much time it takes, just how hard it makes the students think. 8-9. (We're trying to get a feel for the think/tool ratio here.)

4. On what basis is the grade for the subject determined (for instance, quizzes, problem sets, labs, participation)? Is there an equation that will suffice for this answer?

4 Quizes each equal weight – 50-60%

Problem sets 40-50%

5. Please comment on the the teaching philosophy of the course. For instance, are problem sets and labs intended to reinforce material already taught or to lead students to discover new results? Are quizzes similar to or different from the problem sets in terms of content and difficulty? Problem sets are designed to reinforce material taught in lectures, but often they direct students to ~~[illegible]~~ filling ~~out~~ details left out in lecture, or exploring directions we did not persue in lecture. Quizes are similar to problem sets in terms of content, but are designed to be easier than problem sets with the quiz question giving more structure and detailed guidance than a comparable problem set problem.

6. What recommendations do you have for students taking the class or considering taking the class? What are students doing right, and what are they doing wrong this term?

This class has seemed more passive during lectures than in previous years. A lecture w/ 15-20 should be able to be interactive. We have found it very difficult to get feedback from students during class — Did they understand this definition? Is this proof convincing? Why do we care? Why did the lecturer say 2+2=7? These are questions students must have, but are not voicing. This makes it difficult to properly pace the lectures and gauge the level of difficulty.

7. Do you have any responses to expected criticism of the subject? Do you have any criticism of your own?

8. What textbook(s) are used? Please give the authors as well. Are they available anywhere other than the Coop?

The textbook is a photocopyed draft of Introduction to the Formal Semantics of Programming Languages by Glynn Winskel. It was sold to students at cost by the course staff.

9. What changes do you suggest for the next time the class is taught? Will you be teaching the class again in the future (in particular, the next time it is offered?)

The class will be taught by A. Meyer the next time it is taught. It will be more similar to how it was this term rather than how it was done in past years. Two changes however, should be a more detailed syllabus and notes for lectures which were not based on text will be distributed in advance. In addition, there will be a somewhat different introduction to logic, and the fundamental notions of soundness and completeness.

10. What facet of the course (if any) do you think could be changed to make the students focus on understanding the material instead of what grade they receive? Would you be willing to make such a change(s)?

?

11. If student evaluations arrived on your desk next term, would you _____ pitch them, _____ distribute them to staff but not read them yourself, _____ read them time permitting and then distribute them to staff, __X__ read them all and then distribute them to staff?

12. Please use this space for general comments about the course in general. Add any other comments which you feel are appropriate.

Please See attached.

13. We are also very interested in your comments on the *Underground Guide to Course VI* and on this form. How can we make the *Guide* more helpful to you and this form more helpful to everyone?

Thank you for your cooperation.

All 6-3 students are required to take either 6.045 or 6.044. Both are very ~~[illegible]~~ theoritically oriented classes. Both are grounded on basic mathmatical notions and methods of proof which are used for showing certain problems to be "computable" or "uncomputable". Both expect students to do rigorous mathematic proofs. Their approaches to these notions are very different. 6.044 tends to focus on comparing properties of programming languages and proving general properties of programs. 6.044 also provides a heavy exposure to 1st order logic, and Hoare logic which is used for proving properties of programs. 6.045 tends to focus on various models of computation: finite automata, non-deterministic finite automata, ~~and~~ pushdown-automata, and turing machines. 6.045 also provides an introduction to basic notions of computability theory — obscuring distinctions between programming languages, and focusing on the "inherent difficulty of various problems".

6.044 is generally offered in the Fall, 6.045 is generally offered in the Spring. There may be an impression among the students that 6.044 is a harder class than 6.045, this may in fact be true, but it is <u>NOT</u> intentional. Any difference in difficulty is a calibration problem on our part which we may attempt to correct.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044 J

Today's date: 22 NOV 91

Lecturer(s): PROF ALBERT MEYER

Your year: 4

Recitation Instructor:

TA(s): ARTHUR LENT

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

AS FAR AS I CAN TELL, THIS SUBJECT IS ABOUT PROVING OBVIOUS THINGS IN GREAT DETAIL. I DIDN'T ~~[illegible]~~ FEEL THAT I LEARNED A LOT OF NEW USEFUL INFORMATION, BUT I DID LEARN HOW TO PROVE OBVIOUS THINGS, WHICH WAS NOT THAT INTERESTING TO ME SINCE ~~[illegible]~~ I ALREADY BELIEVED THAT THE FACTS WERE TRUE.

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 8.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - ____ - 5.5.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? (YES) Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

PROF. MEYER IS A GOOD, CLEAR LECTURER, BUT HE GOES TOO FAST. HE IS FRIENDLY & HELPFUL & I APPRECIATE HIS MANY REQUESTS FOR FEEDBACK FROM THE CLASS.

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

N/A

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

TA IS VERY HELPFUL & AVAILABLE OFTEN. HOWEVER, IT WOULD BE HELPFUL TO HAVE REGULAR OFFICE HOURS INSTEAD OF "BY APPOINTMENT"

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets _X_ labs ____ lecture _X_ recitations _X_ tutorials _X_ ~~reading~~ ____. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ____ recitations _X_ tutorials ____ ~~reading~~ ____ labs____.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

I DID NOT UNDERSTAND A WORD OF THE TEXT

EVEN AFTER I LEARNED THE MATERIAL FROM LECTURE OR TUTORIAL, I HAD A HARD TIME GOING BACK & READING THE SAME MATERIAL IN THE TEXT.

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

THE SUBJECT WAS COMPLETELY UNINTERESTING TO ME, BUT I HAD TO TAKE IT AS A REQUIREMENT. IT WAS TAUGHT WELL, BUT I'M JUST NOT INTERESTED IN THE MATERIAL AT ALL.

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

14. Did you get enough time to fill out this form? yes ____ no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044J/18.423J

Today's date: 11/22/91

Lecturer(s): Meyer

Your year: 4

Recitation Instructor: ——

TA(s): Arthur Lent

My Course: 6-3

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

Learned about logic and programming. Class was really all theory and no application. A weak point of the subject is we get far too caught up in notation

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

6.035, 6.001, 18.063

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 7.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 9.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

Lecturer is very articulate and I get the feeling he knows what he wants to say, but oftentimes he gets confused. He would probably to better if he wrote out what he wants to cover, but he just "wings it"

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

No rec instructor

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

TA has office hours once a week, but a couple of times when I went to see him he wasn't there. When he is there, though, he answers questions very well and clearly. He has more than enough technical background. He could probably teach the course.

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets X labs ____ lectur X recitations ____ tutorials ____ reading X. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

The quizzes are very confusing, and often have mostly problems with no partial credit. This is very painful considering there are usually only 4 problems per quiz

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures X recitations ____ tutorials ____ reading X labs____.

Problems set are very long, tedious, and complicated. I'm not so sure they have to be so difficult. It would be better if they simply helped us to learn the material.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

No labs

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

Textbook is almost (not quite) as confusing as the letures. Again, we tend to get caught up in notation. A better written textbook (i.e. easier to read) would be better.

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

I haven't quite decided if it's worthwhile. The course is a little too much theory and no application.

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

14. Did you get enough time to fill out this form? yes X no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044 J

Today's date: 22 Nov '?

Lecturer(s): Meyer

Your year: '92

Recitation Instructor: —

TA(s): A. Lent

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

Interpretation of computer programs - lots of logic

Very much theory

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 8.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 6.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

Teaches well, doesn't assume too much - ~~learn about as much from~~ text is good to go back to after he's done lecturing

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

No Recitation.

TA is very cocky. He tries to be helpful, though

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

There were some test review sessions, TA seems to know what he's doing.

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets X labs ____ lectures X recitations ____ tutorials X reading X. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

quizzes are somewhat on the hard side, and very long

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ____ recitations ____ tutorials ____ reading ____ labs____.

they are challenging, material is taught before hand

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

—

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

Textbook is a draft of a text - it's ok

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

Not terribly. Know the stuff cold or else you'll get reamed.

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

Meyer has a bad reputation but (so far) seems like a pretty nice guy.

14. Did you get enough time to fill out this form? yes X no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044J/18.423J

Today's date: 11/2[illegible]

Lecturer(s): Meyer

Your year: Jr.

Recitation Instructor:

TA(s): Arthur Lent

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

Formal program verification

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 6.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 6.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

Somewhat pedantic. Easier to understand than the book. Think formal program verification is the cat's pants so uses bizarre formal verification arguments when you're asking a simple question. Not much good as far as helping us understand when we don't think the same way he does.

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

My TA is wonderful, Friendly, and can explain things, and down to earth and makes sense.
I [illegible] not have made it through the course without [illegible]

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets ✓ labs ____ lectures ✓ recitations ____ tutorials ✓ reading ____. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

The quizzes are very reasonable. The problem sets prepare you well for them

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ✓ recitations ____ tutorials ____ reading ____ labs____.

The problem sets are good. Annoying [illegible] occasional bugs, but good sets. Also the TA + prof are very responsive if we say this question is way too hard and takes too long - they make the question shorter, simpler etc

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

We are using a draft of textbook that is very hard to read. After reading a section about 4 times I get an idea of what they are saying, but I don't think it's a good book

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

I was very intimidated being ~~the~~ one of 2 women in this class and one of 2 non-course six majors
Why are there so few women in this class?

14. Did you get enough time to fill out this form? yes ____ no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044J / 18.423J

Today's date: 11-22-91

Lecturer(s): Meyer 11200

Your year: 4

Recitation Instructor:

TA(s):

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

This is a math class about the meaning of computer programs and formal reasoning techniques about computer programs. I learned nothing about computer programs. I learned quite a bit about foundational mathematics. Generalized axiomatic systems, generalized (Rule) induction, some Set/Function theory.

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

18.063 is reasonable. An enjoyment of abstract ideas.

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 7.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 4.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

The lecturer tries very hard to strip away the formalism that impedes understanding, and often succeeds. The text is followed closely and this is helpful.

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

TA is very good, knows the subject as well as the professor, and is available for questions. No tutorials.

The Structure of the Subject

8. Which of the following ar; essential to do well on the quizzes? problem sets ✓ labs ____ lectures ✓ recitations ____ tutorials ____ reading ____. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

The Quizzes are much easier than the problem sets. Length and grading are very fair (maybe a little short.)

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ____ recitations ____ tutorials ____ reading ____ labs____.

The problem sets are fair but challenging. They follow the lectures well. Sometimes they require excessive symbol-manipulative grungework.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material? I thought the draft of our text (Whiskel) was decent. I have only read it when I get confused, and it has been helpful.

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

Only in a general intellectual sense. I can not imagine it ever being directly useful for me. You must enjoy proving the obvious as an intellectual exercise in order to enjoy this class.

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments. ? I know nothing about this subject other than what I have learned in the class, so I have no idea!

14. Did you get enough time to fill out this form? yes ✓ no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044

Today's date: 11-22-91

Lecturer(s): A. Meyer

Your year: 3

Recitation Instructor:

TA(s): Arthur Lent

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory. A class on the logic and theory of programming. A simple imperative language was used to learn about evaluation, syntax, several kinds of semantics, and reasoning about programs. The simple language makes everything clear, but (practical) application is not stressed.

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)
Mathematical Sophistication

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 8.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 6.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)
Lecture is much clearer than the readings. Prof. Meyer is willing and capable of making everyone understand his lectures. Use of blackboard is good; perhaps too many long proofs.

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background? TA is available by e-mail which is convenient. TA was capable of giving entire lecture by himself! He already knows this stuff inside and out!

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets ____ labs ____ lectures ____ recitations ____ tutorials ____ reading ____. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ____ recitations ____ tutorials ____ reading ____ labs____.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

14. Did you get enough time to fill out this form? yes ____ no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044J

Today's date: 11/22

Lecturer(s): Meyer

Your year: 4

Recitation Instructor:

TA(s): Arthur Lent

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

Prereq: 18.063, love for induction

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 9.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 6.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

The lectures are often straight out of the reading. More example problems in lecture would be helpful. Meyer always goes 10 minutes overtime in lectures, also. We never get out until 2:05. The lectures are generally well organized and clear, though. (except during the last minutes)

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

No recitations.

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

Arthur is very available and is very patient in answering questions. He seems to know at least as much as ~~the~~ Meyer. The quiz reviews he holds are great.

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets _✓_ labs ____ lectures _✓_ recitations ____ tutorials ____ reading _✓_. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

The quizzes are long, but fair They are usually similar to the problem sets. They are also graded very promptly - a definite plus.

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures _✓_ recitations ____ tutorials ____ reading _✓_ labs____.

If you don't do the problem sets, you're hosed. They're really hard, but if you eventually figure them out, you'll be in good shape for the tests.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

No labs.

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

The notes are useful, but terrible about references.

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

14. Did you get enough time to fill out this form? yes _✓_ no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044 J

Today's date: 11/22/91

Lecturer(s): Meyer

Your year: '92

Recitation Instructor: –

My course: 6-3

TA(s): A. Lent

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

Another boring math class required for c.s..

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

None.

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 6.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 2.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

Lecturer is fine, although he ~~tends~~ has a tendency to lose his train of thought and never find it – often very confusing.

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

—

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

TA was available and willing to help.
No tutorials, but review sessions were helpful.
Yes, he seems to know the subject well.

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets _no_ labs ___ lectures _a little_ recitations ___ tutorials ___ reading _yes_. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

Problem sets were sometimes difficult, but not too many or too long.

Quizzes were perfect - not too long or hard - tested your actual knowledge.

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ____ recitations ____ tutorials ____ reading ____ labs____.

Yes, material is taught beforehand. Yes, they are challenging.
No, they are not essential to learning the material.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

—

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

Book (notes) is unfinished and needs editing, but it is very helpful for learning the course.

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

I wouldn't take this class if i didn't have to.

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

General comment: This was possibly the best organized and administered class i've ever been in.

14. Did you get enough time to fill out this form? yes _X_ no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044J

Today's date: 11/18/91

Lecturer(s): Meyer, Albert

Your year: 4

Recitation Instructor: /

TA(s): Arthur Lent

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

I have learned some interesting material on the mathematical foundation of languages. The class is interesting but I don't ever see myself applying much if not any of the material in the real world. I now know induction in 31 flavors.

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

18.063 and good background in proofs.

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 7. and 10 if you really want to understand

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 11.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

The lectures help understand the cryptic text that has cryptic examples. However, the lecturer doesn't always come fully prepared and you can walk out of class with a big ?

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

No recitation.

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

The TA was great. He is the one who has taught me most of the material. I liked his lectures better than the main lecturer. Prof. Meyer assumes too often that we all really do understand what he is saying.

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets X labs X lectures ____ recitations ____ tutorials X reading ____. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

The quizzes are fairly representative. The [illegible]

problem sets are not typically really difficult. once you understand them. otherwise it might take 4 hours to do one problem

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ____ recitations X tutorials X reading ____ labs____.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

N/A

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

The notes (textbook) are rather concise but they are too cryptic to read if you don't have a strong proof-oriented background.

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

Ok but I wouldn't take just for my own enjoyment ... it fills a requirement.

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

The lectures just need to be clearer and planned out more ahead of time

14. Did you get enough time to fill out this form? yes X no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044

Today's date: 11/22

Lecturer(s): Meyer

Your year: 4

Recitation Instructor: Arthur Lent

6-3

TA(s):

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

Learn how to prove an overwelming amount of statement's about the math underlying programming languages

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

a lot of theorectical math subjects, patience, and the ability to take [illegible]

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 9.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 9.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

Lecturer does a good job, but could benefit from more preparation on the "hairy" topics. Lecturer improved on his "intuitively obvious" assumptions.

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

ARTHUR DOES A GREAT JOB ORGANIZING THE CLASS (BOTH IN CLASS AND ON ATHENA FORUM), GRADING [illegible] AND [illegible] THUMBS [illegible]

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets ✓ labs ____ lectures ✓ recitations ____ tutorials ____ reading ✓. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

QUIZZES ARE DIFFICULT, BUT NOT IMPOSSIBLE. THEY TEND TO BE TOO LONG (DUE TO THE AMOUNT OF THOUGHT REQUIRED FOR EACH PROBLEM) THEY ARE GRADED ALL-OR-NOTHING STYLE.

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ✓ recitations ____ tutorials ____ reading ✓ labs____.

The problem sets are <u>too</u> hard. They test your ability to do <u>too</u> much <u>too</u> fast. They should be a little less challenging and a little more helpful to gain a solid background, instead of a headache.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

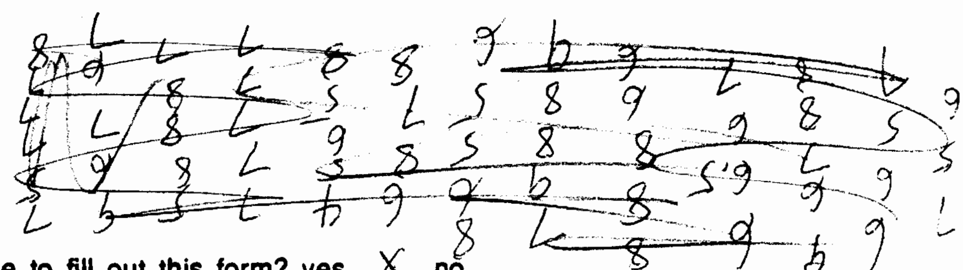
11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

TEXTBOOK IS O.K FOR A MATH THEORY COURSE. IT COULD STAND A FEW MORE EXAMPLES AND A LITTLE BETTER EXPLANATION TO US MORONS!

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

I DON'T SEE ANY REASON THAT I WILL EVER APPLY THE KNOWLEDGE (OTHER THAN BASIC MATH LOGIC) IN MY LIFE.

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.



14. Did you get enough time to fill out this form? yes X no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044J, 18.423J

Lecturer(s): Prof. Meyer

Recitation Instructor:

TA(s): Arthur Lent

Today's date: 11/22/91

Your year: 92

Course 6-3

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

Formal Semantics of programming languages. Mostly a theoretical course, but with valuable applications.

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 9.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3 - 0 - 15.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

Lecturer was probably very good- things got to intimidating for me.

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

TA is one of the best I've had in my 4 years. Knows everything cold. No tutorials, TA available for extra help.

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets X labs ____ lectures X recitations ____ tutorials ____ reading X. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

All quizzes were fair, and graded fairly. Quiz 3 was long, but ample time was given (extra 1:05). Problem sets are difficult.

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ____ recitations ____ tutorials ____ reading ____ labs____.

Problem Sets are very relevant

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

Textbook was a draft of an unpublished book. Excellent guide.

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

If you want to learn this stuff, this is where to get it.

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

14. Did you get enough time to fill out this form? yes X no____.

The Structure of the Subject

8. Which of the following are essential to do well on the quizzes? problem sets ✓ labs ____ lectures ____ recitations ____ tutorials ____ reading ✓. Comment on the length, difficulty, grading, etc. of the quizzes, and compare them to the problem sets and labs.

9. Comment on the problem sets. Is the material taught before you have to do them? Are they helpful in learning the subject matter? Do they challenge you? What is essential to do the problem sets? lectures ✓ recitations ____ tutorials ____ reading ✓ labs____.

10. What is your opinion of the labs and how can they be improved? Comment on equipment quality, availability, accessibility, etc. How helpful is the lab staff?

11. Comment on the textbook(s)/class notes. How could they be improved? Are they useful in learning the material?

12. Is the class worthwhile? What advice would you give friends planning to take this subject?

13. What should have been taught that wasn't? What material should be dropped? Use this space for any general comments.

14. Did you get enough time to fill out this form? yes ____ no____.

Eta Kappa Nu Subject Evaluation Form

Return completed form to **38-476** before 5pm today

Subject Number: 6.044

Today's date:

Lecturer(s):

Your year: 4

Recitation Instructor:

TA(s):

1. Briefly describe the subject. What did you learn? What are the subject's strong and weak points? Also comment on the balance between application and theory.

2. What should be the prerequisites and corequisites for the subject? (They don't have to be subjects.)

Know how to do proofs

3. Rate the difficulty of this subject on a scale of 0 to 10, where 0=trivial, 5=average MIT, 10=painful. This is not asking how much time it takes, just how hard it makes you think! 8.

4. How much time does this subject *actually* require? Give the standard weekly class-lab-prep hours. Remember that a 6-hour lab every other week averages to 3 lab hours. 3-0-10.

The Quality of the Teaching

5. Comment on the lecturer's teaching. Are you learning more in lecture than from the reading? Does the lecturer follow a syllabus? Does the lecturer assume too much as being "intuitively obvious?" Also comment on the lecturer's presentation methods (use of blackboard, transparencies, handouts, demos, etc.)

TEXT - LECTURES very similar

6. Comment on the recitation instructor's teaching. How do the recitations benefit you? If they don't, how can they be improved?

7. Comment on your TA's availability and willingness to answer questions. Are there tutorials, and do you attend them? Does your TA have enough technical background?

Good TA

Solutions to Diagnostic Quiz

Problem 1. Describe the function which is the composition of the integer successor function, *i.e.*, successor$(x) = x + 1$, with itself. Answer: $x + 2$.

Problem 2. How many strings of length four are there over the alphabet $\{a, b, c\}$? Answer: $3 * 3 * 3 * 3 = 81$; for each position there are three possible letters, and there are 4 possible positions.

Problem 3. Give an example of an uncountable set. Examples: the real numbers, and the real numbers between 0 and 1.

Problem 4. Which is a synonym for "injective"? Answer: (e) one-to-one.

What sets have the property that there is *no* injection from the set into itself? Answer: NONE. The identify function from a set onto itself is always well-defined, and always an injection.

What sets have the property that there is *no* injection from the set into a *proper* subset of itself? Answer: Precisely the finite sets.

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \rightarrow B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a counterexample.

- **(a)** reflexive. Answer: YES. The identity from A to A always exists and is always injective.
- **(b)** symmetric. Answer: NO. Consider $A = \{1\}$ and $B = \{1, 2\}$. $A \preceq B$ but $B \not\preceq A$.
- **(c)** transitive. Answer: YES. If f_1 is an injection from A to B and f_2 is an injection from B to C then $f_2 \circ f_1$ is an injection from A to C.
- **(d)** equivalence relation. Answer: NO. A relation is an equivalence relation iff it is reflexive, symmetric and transitive. $\preceq$ is not symmetric.
- **(e)** partial order. Answer: No. A relation is a partial order iff it is reflexive, transitive, and ***anti-symmetric***, *i.e.*, if A is related to B and B is related to A then $A = B$. If we consider the case of $A = \{1, 2\}$ and $B = \{3, 4\}$, then $A \preceq B$ and $B \preceq A$, but $A \neq B$.

Problem 6. Describe a propositional, *i.e.*, Boolean, connective which is not commutative. Answer: Implies ($\supset$) is a propositional connective which is not commutative. (8 of the 16 propositional connectives are not commutative).

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are *equivalent* iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables. Example: $x_1 \supset x_2$, $\overline{x_1} \vee x_2$ and $\overline{x_1} \vee x_2 \vee x_2$ are true for all assignments *except* $x_1 = \mathtt{true}$ and $x_2 = \mathtt{false}$, in which case all are false.

(b) Explain why "equivalence" is actually an equivalence relation on formulas. Answer: Because it is reflexive (obviously), symmetric (if F_2 agrees with F_1 on all input values, then the opposite must also be the case), and transitive (if F_1 agrees with F_2 on all inputs values, and F_2 agrees with F_3 on all input values, then F_1 agrees with F_3 on all input values), by definition the relation "equivalence" is an equivalence relation on formulas.

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many? Answer: For n variables there are exactly 2^n different 0-1 assignments to the variables. For each assignment to the variables there are two possible truth values to yield. Consequently there can be at most only 2^{2^n} different equivalence classes. Why? By the pigeonhole principle if there were more than this 2^{2^n} equivalence classes then at least two of them would have to have the same input/output behavior, in which case they would be the same equivalence classes, so there can be at most 2^{2^n} distinct equivalence classes.

Chris K. Brown
9/13/91
6.044 Diagnostic Quiz

—Begin 6:54 PM

1) $s(x) = x+1$
✓ $s(s(x)) = s(x+1) = (x+1)+1 = x+2$

2) 3 choices, each of 4 times
✓ $3 \cdot 3 \cdot 3 \cdot 3 = 3^4 = 81$ strings of length 4.

3) ✓ The set R of Real numbers.

4) I don't know what 'epi' means. The one the looks ~~like~~ a good possibility is (b) onto, Since one set will map onto the other.

x Not to sure about the first one. Infinite sets, maybe?

✓ Finite sets cannot have an injection from the set into a proper subset of itself.

5. Don't know about binary relations. I know about binary operations and groups, though

6. Never heard the term 'Boolean Connective'

7. a) $x_1 x_2$
x $\neg(x_1 + x_2)$ $(\neg x_1)(\neg x_2)$
$(x_1 x_1)(x_2 x_2)$

b) An 'equivalence relation' is a relation that is reflexive, symmetric and transitive. These 'equivalent' formulas have
x the same properties.

c) I wont attempt to tackle that.

Finished 7:43.
Elapsed - 49 minutes.

6.044 Diagnostic Quiz

Author: Ken Duda
Date: 9/11 4:30 pm

Problem 1

$$f(f(x)) = (x+1) + 1 = x + 2$$

Problem 2

$$3^4 = 81$$

Problem 3

The real numbers, $\Re$, are not countable.

Problem 4

Injections are one-to-one (e).

No sets. There exists at least one injection from any set onto itself, namely, Id, or any other permutation.

All sets. There is no way to have an injection from a larger set to a smaller set.

Problem 5

a — reflexive: yes, $A \preceq A$

b — symmetric: no. $A = \{1\} \wedge B = \{2,3\} \rightarrow (A \preceq B) \wedge (B \not\preceq A)$

c — transitive: yes, $A \preceq B \wedge B \preceq C \rightarrow A \preceq C$

d — equivalence relation: no, not symmetric.

e — partial order: maybe. It does "order" sets according to their cardinality; $A \preceq B \leftrightarrow$ [redacted] $|A| \leq |B|$

Problem 6

$<$ is boolean and doesn't commute; is it a connective?

binary, not boolean.

Problem 7

a — $F_1(a,b) = a$
$F_2(a,b) = a \cdot b + a \cdot \bar{b}$
$F_3(a,b) = b \cdot \bar{b} + a(b + \bar{b}n)$

b — Let $\cong$ denote functional equivalence. It is clearly reflexive (every $F \cong F$). It is symmetric; if $F_1 \cong F_2$, then for every combination of x_i:

$$
\begin{aligned}
f_1(x_i, \ldots) &= f_2(x_i, \ldots) \\
f_2(x_i, \ldots) &= f_1(x_i, \ldots) \\
f_2 &\cong f_1
\end{aligned}
$$

By the same reasoning, $\cong$ is transitive and is thus an equivalence relation.

c — There are only as many equivalence classes as non-equivalent functions. Consider a function of n variables, each of which can take on v values, into a range of r values. There are only v^n distinct input combinations, each of which can result in only one of r outputs. Therefore, there are

$$v^{(r^n)}$$

total possible functions; in this case, where $v = 2$ and $r = 2$, the number of distinct functions (equivalence classes) is

$$2^{2^n}$$

Michael Fan
9/12/91

6.044 Diagnostic Quiz   Time: 40 minutes

Problem 1
You've stumped me.

✓ Problem 2
$3^4 = 81$

Problem 3
X The set of integers is uncountable   No. I can count them as follows: 0, 1, -1, 2, -2, ...

Problem 4
e)
✓ All sets have an injection into itself.
No sets have an injection into a proper subset of itself.
What about integers. enumeration above shows function from $\mathbb{Z}$ to $\mathbb{N}$ + $\mathbb{N} \subsetneq \mathbb{Z}$

Problem 5
My 18.063 needs refreshing, it seems.

✓ Problem 6
$A \Rightarrow B$

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

✓ Problem 7
a) $A \vee B$
$A \vee (A \vee B)$
$(A \wedge B) \vee (A \vee B)$

b)

✓ c) A formula with n variables can have 2^n inputs. For each complete set of inputs, there is a set of outputs. There are $(2)^{2^n}$ equivalence classes possible.

Mark Haseltine
9/13/90
6.044 Diagnostic Quiz

Time: about 1 hour

Problem 1

✓ composition of successor function with itself

successor (x) = x + 1

successor (successor (x)) = (x+1) + 1 = $\boxed{x + 2}$

Problem 2

How many strings of length 4 are there over $\{a,b,c\}$?

✓

$3 \cdot 3 \cdot 3 \cdot 3 = 3^4 = \boxed{81}$

(annotations: # of choices for first position; # for second; # for third; # for fourth)

Problem 3

✓ Example of an uncountable set

the set $\mathbb{R}$ of real numbers

Problem 4

A synonym for "injective" is

✗ (e) mono

Sets having the property that there is no injection from the set into itself

× Sets with repeated elements (?) ← wouldn't be a set

Sets having the property that there is no injection from the set into a proper subset of itself

× Sets of 0 or 1 element

Problem 5

Properties of $\preceq$ where $A \preceq B$ iff $(\exists f: A \to B)(f \text{ is injective})$

✓ (a) Reflexive - Yes

✓ (b) Symmetric - No

A: {1, 2, 3}   B: {1, 2, 3, 4, 5}

Since B has more elements, $A \preceq B$, but $B \npreceq A$

✓ (c) Transitive - Yes

✓ (d) Equivalence relation - No

To be an equivalence relation it has to be reflexive, transitive, and symmetric

✓ (e) Partial order - No

A: {1, 2, 3}, B: {1, 4, 9} are not anti-symmetric

Problem 6

A propositional connective which is not commutative

Less than $(A < B) \neq (B < A)$ (?)

Problem 7

(a) 3 simple, syntatically distinct, but equivalent formulas - 2 variables

$\sim(A \vee B)$, $\sim A \wedge \sim B$, $\sim A \wedge (A \vee \sim B)$

(b) "equivalence" is actually equivalence relation

Must have reflexive, symmetric and transitive properties

Reflexive

$\sim(A \vee B) = \sim(A \vee B)$ True

Symmetric

$\sim(A \vee B) = (\sim A \wedge \sim B) \Rightarrow (\sim A \wedge \sim B) = \sim(A \vee B)$

True

Transitive

$(\sim(A \vee B) = (\sim A \wedge \sim B)) \wedge ((\sim A \wedge \sim B) = (\sim A \wedge (A \vee \sim B)))$
$\Rightarrow (\sim(A \vee B) = (\sim A \wedge (A \vee \sim B)))$

True

(c) Only a finite number of equivalence classes of formulas with variable $x_1, \ldots, x_n$

Since formulas can only evaluate to 0 or 1, there can be only 2 equivalence classes (?)

DAVID HAU 9/12/91 20 minutes

Diagnostic Quiz

You will not be graded on this quiz. Do not discuss it with anyone before taking it. Take it sometime after class, and return it to the TA on Friday, September 13. Be sure to indicate your name, the date, "6.044 Diagnostic Quiz", and the time it took you, on your answer sheet.

Problem 1. Describe the function which is the composition of the integer successor function, *i.e.*, successor(x) = $x + 1$, with itself.

✓ successor(successor(x)) = (x+1)+1 = x+2

Problem 2. How many strings of length four are there over the alphabet $\{a, b, c\}$?

✓ $3^4 = 81$

Problem 3. Give an example of an uncountable set.

✓ $\mathbb{R}$

Problem 4. Which is a synonym for "injective"?

- **(a)** epi
- **(b)** onto
- **(c)** mono
- **(d)** isomorphism
- **(e)** one-to-one
- **(f)** one-to-one and onto

bijective

What sets have the property that there is *no* injection from the set into itself?

What sets have the property that there is *no* injection from the set into a *proper* subset of itself?

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \rightarrow B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a countererxample.

(a) reflexive Yes

(b) symmetric No. e.g. $\mathbb{N} \preceq \mathbb{R}$ but $\mathbb{R} \npreceq \mathbb{N}$

(c) transitive Yes

(d) equivalence relation No, because not all of (a), (b), (c) are true.

(e) partial order

Problem 6. Describe a propositional, *i.e.*, Boolean, connective which is not commutative.

x_1	x_2	$f(x_1,x_2)$
0	0	0
0	1	1
1	0	0
1	1	1

e.g. $f(0,1) = 1$, $f(1,0) = 0$

$\therefore f(0,1) \neq f(1,0)$

not commutative

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are *equivalent* iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables. x_1x_2 ; $x_1x_2 + x_1x_2$; $x_1x_1x_2x_2$

(b) Explain why "equivalence" is actually an equivalence relation on formulas. See below

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many?

(b) ① reflexive ∵ a formula is equivalent to itself.
② symmetric ∵ if F_1 and F_2 are equivalent, then F_2 and F_1 are equivalent.
③ transitive ∵ if F_1 and F_2 are eq. and F_2 and F_3 are eq., then F_1 and F_3 are eq.

Scott Hirayama
September 13, 1991
6.044 Diagnostic Quiz

Time: 1hr.

Problem 1

function which is the composition of the integer successor function with itself

Thes function is basically an incrementor which keeps adding an integer value by 1 each time the function is called.

✗

Problem 2

✓ Strings of length four over alphabet {a, b, c}

$3^4 = 81$ strings

Problem 3

✓ an uncountable set = ℝ (set of real numbers)

Problem 4

Synonym for "injective" => mono (c)

✗ every y in Y is a value f(x) for at most one x in X.

✗ No injection from the set into itself => {∅}
→ does this mean no sets, or only the empty set.

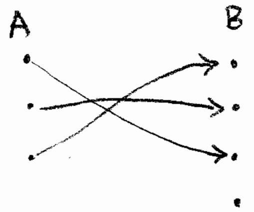
✗ No injection from the set into a proper subset of itself =>

Problem 5 Define a binary relation, ≤, between sets A & B

$A \leq B$ iff $(\exists f : A \to B)$ (f is injective)

$A \subseteq B \iff \forall z \in A . z \in B$

Eg.



Which of the following properties does the relation ≤ have ?> (d) equivalence relation

* ~~Not~~ able to give counter example.

Problem 6

Problem 7

(a)

(b) Equivalence is an equivalence relation on formulas because it is through formulas that we are given a set of instructions on how to arrive at a solution. From the basis of a formula, we can examine the relation they have with other formulas.

(c) there are only a finite number of equivalence classes of formulas w/ variables $x_1, \ldots, x_n$ because for certain variables the equivalence relation does not hold; whether it be a ~~t~~ reflexive, transitive, or symmetric failures.

How many? Good question??

Norman W Koon
9/13/91
6.044 Diagnostic Tes
15 minutes

1.

2. Infinite

3. Integers

4

5.

6.

7. a) $x_1 + x_2$, $\overline{\overline{x_1} \cdot \overline{x_2}}$, $(x_1 \oplus x_2) + (x_1 \cdot x_2)$

b)

c) Because there are only a certain number of possible input combinations which must only give either a 1 or 0 output.

2^{2^n}

SHEUNG LI
Sep 12
6.044 DIAGNOSTIC QUIZ
30 MINUTES

PROBLEM 1. ✓ $f(x) = x+1$

$f \circ f(x) = x+2$

PROBLEM 2. ✓ $\{a, b, c\}$ strings of length 4

$3 \cdot 3 \cdot 3 \cdot 3 = 3^4 = 81$

PROBLEM 3. X THE TIME INTERVALS BETWEEN THE EMISSIONS OF A QUASAR GIVEN A SINGLE FIXED OBSERVER. (THESE ARE INCREASING IN LENGTH AND SEEMINGLY RANDOM)

~~Yes, but only because this~~ No. Since it can be counted; just by mapping 1st interval to 1, 2nd to 2 ⋮

PROBLEM 4. ✓ ONTO

X NO INJECTION TO ITSELF: SETS OF TWO OR MORE, ALL DUPLICATE ELEMENTS

✓ NO INJECTION TO PROPER SUBSET: SETS OF THREE OR MORE, ALL DUPLICATE ELEMENTS

PROBLEM 5.

X $\preceq$ IS NOT REFLEXIVE X NOT SYMMETRIC

TRANSITIVE X NOT EQUIVALENCE RELATION

? PARTIAL ORDER

REFLEXIVE? SUPPOSE $A: \{1, 1\}$

$A \preceq A$ requires $(\exists f: A \to A)$ (f injective)

THERE IS NO f WHICH MAPS AT ANY ONE ELEMENT OF A TO ANY GIVEN ELEMENT OF ITSELF

PROBLEM 5 (CONT.)

SYMMETRIC ? SUPPOSE A: {1, 4}
B: {2, 2}

AT MOST, ONE ELEMENT OF A EXISTS SUCH THAT $f(a) = b$ FOR ANY GIVEN b. BUT CLEARLY, GIVEN ANY ELEMENT OF A, THERE IS MORE THAN ONE b WHICH WOULD SATISFY $f(b)=a$

EQUIVALENCE ? MUST BE REFLEXIVE, SYMMETRIC, AND TRANSITIVE TO BE AN EQUIVALENCE RELATION

PROBLEM 6. √ AND - NOT

	AND-NOT
0 0	0
0 1	0
1 0	1
1 1	0

} NOT COMMUTATIVE

PROBLEM 7.

a) ① (A OR B) OR (A NOR B)

② (A AND B) OR (A NAND B)

③ (A XOR B) OR (A) OR (NOT B)

	①	②	③
0 0	1	1	1
0 1	1	1	1
1 0	1	1	1
1 1	1	1	1

b) "EQUIVALENCE" IN THIS SENSE EXHIBITS REFLEXIVITY, SYMMETRY, AND THE TRANSITIVE PROPERTY. IF THE SAME TRUTH TABLE IS YIELDED REFLEXIVITY AND SYMMETRY ARE READILY APPARENT. SINCE COMBINED TWO TRUTH TABLES IS THE SAME AS "AND"ING THEM AND "AND" IS TRANSITIVE, THIS PROPERTY IS QED.

PROBLEM 7. (CONT.)

c) THERE ARE ONLY A FINITE NUMBER OF EQUIVALENCE CLASSES WITH AT MOST n VARIABLES, BECAUSE THERE IS ONLY A FINITE NUMBER OF TRUTH TABLES GIVEN n VARIABLES. THIS NUMBER IS $2^{\wedge}$

Eliseo Martinez
9/12/91

time: 45 minutes

6.044 Diagnostic Quiz

Problem 1

successor(x) = x+1

successor(s) = successor(x) + 1

✓ f(x) = x + 2

Problem 2

✓ 81

Problem 3

✓ The set of real numbers ($\mathbb{R}$) is an uncountable set.

Problem 4

a synonym for injective is

✓ c) one to one

X Finite sets of integers have no injection from the set onto itself, I think?

Problem 5

a) reflexive implies ($\exists F : A \to A$) (f is injective)
✓ true for the identity function

b) symmetric implies ($\exists F : B \to A$) (f is injective)
X true for the identity function

c) transitive implies $\exists F : A \to B$ (F injective) and $\exists F : B \to C$ (F is injective)
✓ $\Rightarrow \exists F : A \to C$ F is injective

d. All three properties hold therefore there is an
X equivalence relation

ALEJANDRO MEDINA
30 mins.

(*) I couldn't do anything else without going back to brush up on some stuff.

Diagnostic Quiz

You will not be graded on this quiz. Do not discuss it with anyone before taking it. Take it sometime after class, and return it to the TA on Friday, September 13. Be sure to indicate your name, the date, "6.044 Diagnostic Quiz", and the time it took you, on your answer sheet.

Problem 1. Describe the function which is the composition of the integer successor function, *i.e.*, successor$(x) = x + 1$, with itself.

Problem 2. How many strings of length four are there over the alphabet $\{a, b, c\}$?

✓ 3^4

Problem 3. Give an example of an uncountable set.

Problem 4. Which is a synonym for "injective"?

(a) epi

(b) onto

(c) mono

(d) isomorphism

✓ **(e)** one-to-one

(f) one-to-one and onto

What sets have the property that there is *no* injection from the set into itself?

What sets have the property that there is *no* injection from the set into a *proper* subset of itself?

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \to B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a countererxample.

(a) reflexive

(b) symmetric

(c) transitive

(d) equivalence relation

(e) partial order

Problem 6. Describe a propositional, *i.e.*, Boolean, connective which is not commutative.

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are *equivalent* iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables.

(b) Explain why "equivalence" is actually an equivalence relation on formulas.

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many?

(a) $x_1 \wedge x_2$

$\neg(\neg x_1 \vee \neg x_2)$

$(x_1 \wedge x_2) \wedge x_1$

William D. Jinerson 9/15
Time: 20 min
Note: Had to review some 18.063 notes.

e-mail wdj@medg.lcs.mit.edu

Diagnostic Quiz

You will not be graded on this quiz. Do not discuss it with anyone before taking it. Take it sometime after class, and return it to the TA on Friday, September 13. Be sure to indicate your name, the date, "6.044 Diagnostic Quiz", and the time it took you, on your answer sheet.

Problem 1. Describe the function which is the composition of the integer successor function, *i.e.*, $\text{successor}(x) = x + 1$, with itself.

Problem 2. How many strings of length four are there over the alphabet $\{a, b, c\}$?

3^4

Problem 3. Give an example of an uncountable set.

Decimal numbers (real)

Problem 4. Which is a synonym for "injective"?

(a) epi

(b) onto

(c) mono

(d) isomorphism

(e) one-to-one

(f) one-to-one and onto

What sets have the property that there is *no* injection from the set into itself? None

What sets have the property that there is *no* injection from the set into a *proper* subset of itself? ?

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \rightarrow B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a countererxample.

(a) reflexive

(b) symmetric

(c) transitive

(d) equivalence relation

(e) partial order

Problem 6. Describe a propositional, *i.e.*, Boolean, connective which is not commutative.

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are ***equivalent*** iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables.

(b) Explain why "equivalence" is actually an equivalence relation on formulas.

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many?

Diagnostic Quiz

You will not be graded on this quiz. Do not discuss it with anyone before taking it. Take it sometime after class, and return it to the TA on Friday, September 13. Be sure to indicate your name, the date, "6.044 Diagnostic Quiz", and the time it took you, on your answer sheet.

Problem 1. Describe the function which is the composition of the integer successor function, *i.e.*, successor(x) = $x + 1$, with itself.

Problem 2. How many strings of length four are there over the alphabet $\{a, b, c\}$?

Problem 3. Give an example of an uncountable set.

Problem 4. Which is a synonym for "injective"?

(a) epi

(b) onto

(c) mono

(d) isomorphism

(e) one-to-one

(f) one-to-one and onto

What sets have the property that there is ***no*** injection from the set into itself?

What sets have the property that there is ***no*** injection from the set into a ***proper*** subset of itself?

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \rightarrow B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a countererxample.

(a) reflexive

(b) symmetric

(c) transitive

(d) equivalence relation

(e) partial order

Problem 6. Describe a propositional, *i.e.*, Boolean, connective which is not commutative.

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are ***equivalent*** iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables.

(b) Explain why "equivalence" is actually an equivalence relation on formulas.

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many?

MICHAEL D. POWERS
12 SEP 91

6.044 DIAGNOSTIC QUIZ

1) $f(x) \circ f(x) = f(x+1) = ((x+1)+1) = (x+2)$

2) $3^4 = 81$

3) THE SET OF REAL NUMBERS BETWEEN 1 AND 2 (FOR EXAMPLE)

4) (A) ONE-TO-ONE
ALL SETS HAVE AN INJECTION ONTO ITSELF.

ANY FINITE SET HAS NO INJECTION FROM THE SET INTO A PROPER SUBSET OF ITSELF. INFINITE SETS DO.

5) B) NOT SYMMETRIC. FOR THE SETS $A = \{x_1, x_2, x_3\}$ $B = \{y_1, y_2, y_3, y_4\}$
THERE IS AN INJECTION FROM A TO B BUT NOT FROM B TO A

A) YES, REFLEXIVE. THERE IS AN INJECTION FROM A TO A.

C) YES, TRANSITIVE

D) $\preceq$ IS NOT AN EQUIVALENCE RELATION BECAUSE IT IS REFLEXIVE AND TRANSITIVE, BUT NOT SYMMETRIC.

E) I DON'T KNOW.

6)

7 A) A AND B
¬A OR ¬B

Denise A Purdie
9/13/91
~~taken~~ 9/13/91

6.044 Diagnostic Quiz

1. ✓ sucessor (sucessor (x)) = ((x+1)+1) = x+2

2. ✓ 3·3·3·3 = 81

3. ✓ the set of real numbers

4. ✓ (e) one-to-one
(a.)

✓ 5. no finite set is injective into a proper subset of itself.

5. b) ✓ not ~~reflexive~~ symmetric, ie if A={1,2} and B={1,2,3} then B is not injective into A
d) ✓ not an equivalence relation, because it is not symmetric
the relation does have the properties of reflexive, transitive, and partial order

Not a partial order - need anti-symmetry

6 NOT is not commutative, ie A¬B ≠ B¬A
? ¬ is a unary not binary operator.
A¬B does not make sense as a formula

7. a)

6.044 Diagnostic Quiz

Pete Rauch
9-12-91

1. √ successor(successor(x)) = x+2

2. √ 3^4 = 81

3. √ irrational numbers

4. f ?

5. /

6. XOR No

7. (a) √ $A \vee B$

$\neg(\neg A \wedge \neg B)$

$A \vee (B \wedge \neg A)$

(b) Logic is deterministic. ?

(c) There are 2^n possible inputs, and each input into a given formula yields 0 or 1. The "answer for every input" defines an equivalence class. There are $2^{(2^n)}$ such classes.

Time taken: 11 minutes

MICHAEL G. SHELDON

<10 mins

Diagnostic Quiz

You will not be graded on this quiz. Do not discuss it with anyone before taking it. Take it sometime after class, and return it to the TA on Friday, September 13. Be sure to indicate your name, the date, "6.044 Diagnostic Quiz", and the time it took you, on your answer sheet.

Problem 1. Describe the function which is the composition of the integer successor function, *i.e.*, successor$(x) = x + 1$, with itself.

√ f = successor • successor   f(x) = x+2

Problem 2. How many strings of length four are there over the alphabet $\{a, b, c\}$?

√ $3^4 = 81$

Problem 3. Give an example of an uncountable set.

X set of prime #'s

Problem 4. Which is a synonym for "injective"?

(a) epi

(b) onto

(c) mono

(d) isomorphism

(e) one-to-one

(f) one-to-one and onto

What sets have the property that there is *no* injection from the set into itself?

What sets have the property that there is *no* injection from the set into a *proper* subset of itself?

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \to B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a countererxample.

(a) reflexive

(b) ~~symmetric~~

(c) transitive

(d) equivalence relation

(e) partial order

Problem 6. Describe a propositional, *i.e.*, Boolean, connective which is not commutative. ~~discrepency~~ equivals, ≡

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are *equivalent* iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables. $\neg A \vee B$ $\quad A \Rightarrow B \quad B \vee A \equiv A$

(b) Explain why "equivalence" is actually an equivalence relation on formulas.

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many?

Jim Szafranski
6.044 Diagnostic Quiz
9/12 @ 8:00pm

Problem 1

$f(x) = f(x) + 1$

Problem 2

~~2~~ choose 4 with repitition $= \left\langle {3 \atop 4} \right\rangle = \binom{6}{4} = \frac{6!}{4!2!} = 15$

Problem 3

The Set of Real Numbers ($\mathbb{R}$)

Problem 4

i) F (one-to-one & onto)

ii) Null sets

iii) ??

Problem 5

a) TRUE

b) FALSE $A = \{1,2,3\}$ $B = \{1,2,3,4\}$ $A \to B$ $B \not\to A$ ✓

c) TRUE

d) FALSE, because symmetry is not true.

e) TRUE, because $\leq$ possess reflexive, transitive, & anti-symmetry

does not posess anti-symmetry: $A \leq B \wedge B \leq A \Rightarrow A = B$.

Problem 6

any connective requiring order, such as < ("less than").

order on what?

Problem 7

a) i) $\neg x_1 \wedge \neg x_2$ (xor?)   b) ??

ii) $\neg(x_1 \vee x_2)$ (xor?)   c) ??

iii) $\neg((x_1 \wedge x_2) \vee (x_1 \wedge x_2))$

$\neg(x_3 \wedge x_2)$

$\neg x_3 \vee \neg x_2 \neq \neg x_3 \wedge \neg x_2$

Larry Taylor

6.044 Diagnostic Quiz

~ 30 minutes

1.

2. Using one letter, 1
   3 letters. $\underline{3 \times 1 = 3}$

   Using two letters,
   2 and 2 (aabb), 6
   3 and 1 (aaab), 4
   1 and 3 (abbb), 4
   ———
   14

   $\binom{3}{2} = 3$ combinations of 2 letters
   $\underline{3 \times 14 = 42}$

   Using 3 letters,
   (aabc) $\frac{4!}{2!} = 12$
   Each of the three letters could be "the double."
   $\underline{3 \times 12 = 36}$

   ~~2~~ 3 + 12 + 36 = (51)

3. ✓ The set of reals is uncountable.

Benz Theodore
9/13/91
6.044 Diagnostic Quiz
1 hr

Problem 1:
X compose x+1 with itself and describe results.

$f \cdot f = x^2 + 2x + 1$ the composite function is a parabola opening upwards

problem 2:
how many ~~two~~ 4-letter words are there over ✓ the alphabet {a, b, c}?

$3^4 = 81$

Problem 3:
✓ give an example of an uncountable set.
ans. set of real numbers. ℝ.

problem 4:
what is a synonym for "injective"?
X Ans: one to one and onto i.e many to one

What sets have the property that there is no injection from the set into itself?
i.e. what types of sets ~~have no functions~~ for ~~the~~ which no function is able to take a member of the set and produce a ~~membe~~ result which is also a member?

?

What sets have the property that there is no injection from the set into a proper subset of itself?

6.044 Diagnostic Quiz ②

problem 5:
define ~~a~~ binary relation ~~...~~ $\preceq$ s.t.

$$A \preceq B \text{ iff } (\exists f : A \to B)(f \text{ is injective}).$$

let f be: $f(x) = 2x$

since 2x is reflective, symmetric & transitive it is an equivalence relation.

X it does not do a partial order

problem 6: Describe a Boolean Connective which is not Commutative

X $x \wedge y \vee z \neq y \wedge x \vee z$

Commutativity is a property of binary (2-argument) operators. You defined a 3 argument one.

problem 7: Boolean formulas $f_i(x_1, \ldots, x_n)$

a) Using 2 vars.

1. $f_2(x,y) = (x^2 - y^2)^2$ →
2. $f_2(x,y) =$ x XOR y (exclusive or)
3. $f_2(x,y) = (x-y)^2$

f(1)

x	y	
0	0	0
0	1	1
1	0	1
1	1	0

f(2)

x	y	
0	0	0
0	1	1
1	0	1
1	1	0

f(3)

x	y	
0	0	0
0	1	1
1	0	1
1	1	0

b) equivalence is a relation on formulas because it actually tests three properties of formulas. reflexivity, symmetry and transitivity.

c) There are ~~at most~~ only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$.
This is because the equivalence classes each determine a partition of ~~all~~ the whole space.

In this case there are less than ~~n classes~~. or equal to n classes.

6.044 Diagnostic Quiz

Amy Weaver TIME: 1hr~45min. 9/12/91

1) The question seemed vague but here goes

$f \circ f = f(f(x)) = f(x)+1$
$= x+2$

range of f = the domain of f∘f.

$f^{(n)}(f^{(n-1)}(x)) = x+n$

2) ~~108~~ 81 $= 3^4$ (3 alpha, 4 in a string)

or

Table:

Description	Example	Formula	#	× Total Poss	Result
all same	AAAA	4!/3!	4	3	3
1 diff	AAAB	4!/3!	4	6	24
2 diff	AABB	4!/2!2!	6	3	18
3 diff	AABC	4!/2!	12	3	36
					81

3) "Let A be the set of all (possibly infinite) sequences whose elements are the digits 0 and 1. This set A is uncountable."
— from 6.035 Lecture on Sept. 12, 1991 Prof. Gutlag :)

4) a) injective is one-to-one (e)

b) multiple answer
a) null set
b) uncountable set
c) a set w/ no identity operator
d) not surjective

c) ~~uncountable sets~~ all sets

Pg. 1

5) has relation (e) partial order, & maybe (c) transitive - see below.

a) reflexive - no because B is not necessarily A. (ie B can be bigger than A.

b) symmetric - no because B does not necessarily map injectively to A (B could be bigger).

c) transcitive: if $A \leq B$ and $B \leq C$ then $A \leq C$ is true.

d) equivalence relation - no, not onto, B bigger than A

(6) 2 connective but not commutative Boolean:

Unary operator does not even make sense to ask the question →

¬ negation symbol : not

→ conditional symbol : if _ then

7) a) $\neg((A \vee (B \wedge C))$

$\neg((A \vee B) \wedge (A \vee C))$

$(\neg(A \vee B) \vee \neg(A \vee C))$

Sasha Wood
25min

Diagnostic Quiz

You will not be graded on this quiz. Do not discuss it with anyone before taking it. Take it sometime after class, and return it to the TA on Friday, September 13. Be sure to indicate your name, the date, "6.044 Diagnostic Quiz", and the time it took you, on your answer sheet.

√**Problem 1.** Describe the function which is the composition of the integer successor function, *i.e.*, successor(x) $= x + 1$, with itself.

$f(x) = x+1 \quad f \circ f(x) = f(f(x)) = f(x+1) = x+2$

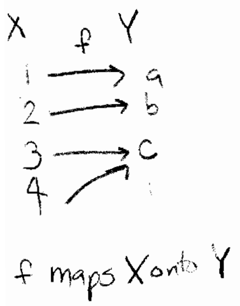
√**Problem 2.** How many strings of length four are there over the alphabet $\{a, b, c\}$?

3^4.

√**Problem 3.** Give an example of an uncountable set.

real numbers
(rational numbers are countable)

Problem 4. Which is a synonym for "injective"?



(a) epi

Note: this is a guess surmising from Probs 4 and 5.

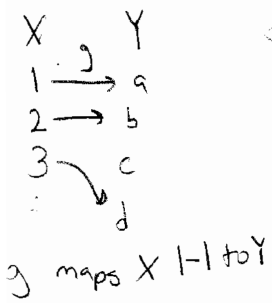
(b) onto

(c) mono

(d) isomorphism

√**(e)** one-to-one

(f) one-to-one and onto



√What sets have the property that there is *no* injection from the set into itself? There are none

What sets have the property that there is *no* injection from the set into a *proper* subset of itself?

only ~~finite~~ All sets ones.

Problem 5. Define a binary relation, $\preceq$, between sets A, B as follows:

$$A \preceq B \quad \text{iff} \quad (\exists f : A \to B)(f \text{ is injective}).$$

Which of the following properties does the relation $\preceq$ have? For those properties it fails, describe some simple sets $A, B, \ldots$ which provide a countererxample.

✓(a) reflexive true

(b) symmetric false $\{1,2,3,4\} = A$ $\{a,b,c\} = B$ $B \preceq A$, but $A \npreceq B$

✓(c) transitive true

✓(d) equivalence relation false $\{1,2,3,4\} = A$. $\{a,b,c\} = B$ A is not equiv. to B.

(e) partial order true

x_1	x_2	
0	0	0
1	0	1
0	1	0
1	1	1

✓**Problem 6.** Describe a propositional, *i.e.*, Boolean, connective which is not commutative. $(x_1 \wedge \overline{x_2}) \vee (x_1 \wedge x_2) = F(x_1, x_2)$

Problem 7. Two Boolean formulas, $F_i(x_1, \ldots, x_n)$ for $i = 1, 2$, are ***equivalent*** iff they yield the same 0-1 truth value for all 0-1 assignments to the variables $x_1, \ldots, x_n$.

(a) Exhibit three simple, syntactically distinct, but equivalent formulas with two variables.

(b) Explain why "equivalence" is actually an equivalence relation on formulas.

(c) Explain why there are only a finite number of equivalence classes of formulas with (at most) variables $x_1, \ldots, x_n$. How many?

a.

x_1	x_2	
0	0	0
1	0	1
0	1	1
1	1	0

✓ $(x_1 \wedge \overline{x_2}) \vee (\overline{x_1} \wedge x_2)$, $(x_1 \wedge \overline{x_2}) \vee (\overline{x_1 \vee \overline{x_2}})$, $(\overline{\overline{x_1} \wedge x_2}) \vee (\overline{x_1} \wedge x_2)$

b. I've forgotten the definition of equivalence relation, but this seems fairly self-evident

✓ c. Because for n variables there are a finite # of truth tables. i.e. There will be 2^n rows in the truth table (2^n possible choices for inputs). So what we want to know then is how many possible choices are there for the output. There are $2^{\# \text{ rows}}$ choices for output — so there are $2^{(2^n)}$ choices for the truth table. And all a formula does is expand a truth table, so there are only $2^{(2^n)}$ formulas and thus only $2^{(2^n)}$ equivalence classes

Conrad Yoder
13 Sept 1991

Diagnostic Quiz 6.044

1)

2) ✓ 3^4 strings

3) ~~X~~the integers

4)

5)

6) ✓an "$\wedge\neg$" function which takes the intersection of the first item with the negation of the second

$1 \wedge\neg 0 = 1$ while $0 \wedge\neg 1 = 0$

7) a) ✓

$$F_1(x,y) = \neg(x \wedge y)$$
$$F_2(x,y) = \neg x \vee \neg y$$
$$F_3(x,y) = \neg((x \vee y) \wedge (x \wedge y))$$

b)

c)

Time Spent: 70 minutes