Questão 1 (Semântica operacional)
A semântica operacional natural (big-step) é uma formalização usando regras de prova. As regras permitem de provar um relação entre o estado inicial, o programa e o estado final: que o programa a partir do estado inicial termina com o dado estado final. Ela é caracterizada por chegar diretamente no estado final, com eventuais pre-requisitos embutidas nas premissas da prova. Em contraste, a semântica operacional estrutural é baseado num sistema de transição que permite multiplas transições para chegar do estado inicial no estado final. Assim, ela mostra os detalhes da execução e é menos abstrato.
A formalizações diferentes também resulta num poder de modelagem diferente. Na aula foi dado o exemplo do paralelismo, que é facilmente à formalizer na semântica operacional estrutural, mas não tem formalização na semântica operacional natural.
Um exemplo de uma regra na semântica operacional natural é

$$\frac{\mathbf{b}, \sigma \Downarrow \text{true}}{\neg\mathbf{b}, \sigma \Downarrow \text{false}}\ \text{not}$$

Exemplos de regras correspondentes na semântica operacional estrutural são

$$\frac{\mathbf{b}, \sigma \rightarrow \mathbf{b}', \sigma}{\neg\mathbf{b}, \sigma \rightarrow \neg\mathbf{b}', \sigma}\ \text{not} \qquad \frac{}{\neg\mathbf{t}, \sigma \rightarrow \mathbf{t}', \sigma}\ \text{not, com } t' = \text{true, se } t = \text{false e false, senão}$$

Questão 2 (Semântica axiomática: Mínimo de três)
A implementação não contém laços, logo as provas do corretude parcial a total são identicas. A execução nesse caso sempre termina.

$\{\ \text{true}\ \}$
$\{\ a < b \rightarrow \Phi \wedge a \geq b \rightarrow \Psi\ \}$
if (a<b) **then** (
  $\{\ a < c \rightarrow a = \min(a, b, c) \wedge a \geq c \rightarrow c = \min(a, b, c)\ \}$
  if (a<c) **then** (
    $\{\ a = \min(a, b, c)\ \}$
    m:=a
    $\{\ m = \min(a, b, c)\ \}$
) **else** (
    $\{\ c = \min(a, b, c)\ \}$
    m:=c
    $\{\ m = \min(a, b, c)\ \}$
)
  $\{\ m = \min(a, b, c)\ \}$
) **else** (
  $\{\ b < c \rightarrow b = \min(a, b, c) \wedge b \geq c \rightarrow c = \min(a, b, c)\ \}$
  if (b<c) **then** (
    $\{\ b = \min(a, b, c)\ \}$
    m:=b
    $\{\ m = \min(a, b, c)\ \}$
) **else** (
    $\{\ c = \min(a, b, c)\ \}$
    m:=c
    $\{\ m = \min(a, b, c)\ \}$
)
  $\{\ m = \min(a, b, c)\ \}$
)
$\{\ m = \min(a, b, c)\ \}$

com as abreviações $\Phi = a < c \rightarrow a = \min(a, b, c) \wedge a \geq c \rightarrow c = \min(a, b, c)$ e $\Psi = b < c \rightarrow b = \min(a, b, c) \wedge b \geq c \rightarrow c = \min(a, b, c)$. A análise dos casos justifique o primeiro passo. Por exemplo: $a < b \rightarrow a < c \rightarrow a = \min(a, b, c) \equiv a < b \wedge a < c \rightarrow a = \min(a, b, c)$ é o primeiro caso que é verdadeiro: dado as duas condições, a é o mínimo de a, b e c.

Questão 3 (Semântica denotational)

(a) Se $y \geq 0$ o programa termina com $y = 0$ e x contém a soma dos valores inicias de x e y.

(b) A equação semântica do laço é $C[\![\texttt{while y}\neq\texttt{0 do x:=x+1;y:=y-1}]\!] = \text{fix}\, F$ com $F(f) = \text{cond}(B[\![y \neq 0]\!], f \circ C[\![\texttt{x:=x+1;y:=y-1}]\!], \text{id})$ e com $C[\![\texttt{x:=x+1;y:=y-1}]\!] = (\lambda\sigma \in \Sigma_\perp.\sigma[x \mapsto \sigma(x)+1][y \mapsto \sigma(y)-1])$ obtemos o funcional

$$F(f)\sigma = \begin{cases} f(\sigma[x \mapsto \sigma(x)+1][y \mapsto \sigma(y)-1]) & \text{se } \sigma(y) \neq 0 \\ \sigma & \text{se } \sigma(y) = 0. \end{cases}$$

e as aproximações

$$F^0(\perp) = \perp$$

$$F^1(\perp)\sigma = \begin{cases} \perp & \text{se } \sigma(y) \neq 0 \\ \sigma & \text{se } \sigma(y) = 0 \end{cases}$$

$$F^2(\perp)\sigma = \begin{cases} \perp & \text{se } \sigma(y) \notin \{0,1\} \\ \sigma[x \mapsto \sigma(x)+1][y \mapsto 0] & \text{se } \sigma(y) = 1 \\ \sigma & \text{se } \sigma(y) = 0 \end{cases}$$

$$F^3(\perp)\sigma = \begin{cases} \perp & \text{se } \sigma(y) \notin \{0,1,2\} \\ \sigma[x \mapsto \sigma(x)+\sigma(y)][y \mapsto 0] & \text{se } \sigma(y) \in \{1,2\} \\ \sigma & \text{se } \sigma(y) = 0 \end{cases}$$

(c) Depois n iterações temos

$$F^n(\perp)\sigma = \begin{cases} \perp & \text{se } \sigma(y) \notin [0,n[\\ \sigma[x \mapsto \sigma(x)+\sigma(y)][y \mapsto 0] & \text{se } \sigma(y) \in [0,n[\end{cases}$$

e o limite superior mínimo dessa cadeia é

$$F^\infty(\perp)\sigma = \begin{cases} \perp & \text{se } \sigma(y) < 0 \\ \sigma[x \mapsto \sigma(x)+\sigma(y)][y \mapsto 0] & \text{se } \sigma(y) \geq 0 \end{cases}$$

Questão 4 (Semântica operacional: Valores "default")

(a) A gramática das expressões aritméticas extendida é

$$\mathbf{e} ::= \cdots | \mathbf{e_1}?\mathbf{e_2} | \cdots$$

(b) Uma solução possíveil na semântica operacional estrutural é

$$\frac{\mathbf{e_1}, \sigma \to \mathbf{e_1'}, \sigma'}{\mathbf{e_1}\texttt{?}\mathbf{e_2}, \sigma \to \mathbf{e_1'}\texttt{?}\mathbf{e_2}, \sigma'} \text{ ?-eval-lhs}$$

$$\frac{}{\texttt{0?}\mathbf{e_2}, \sigma \to \mathbf{e_2}, \sigma} \text{ ?-default}$$

$$\frac{}{\mathbf{n}\texttt{?}\mathbf{e_2}, \sigma \to \mathbf{n}, \sigma} \text{ ?-non-default, com } n \neq 0$$

(c) No sistemas de tipos é suficiente de verificar que as duas sub-expressões são bem tipadas. A regra

$$\frac{\gamma \vdash_A \mathbf{e_1} : \text{int} \qquad \gamma \vdash_A \mathbf{e_2} : \text{int}}{\gamma \vdash_A \mathbf{e_1}\texttt{?}\mathbf{e_2} : \text{int}} \, t\text{–?}$$

cuida disso.

Questão 5 (Questão extra: Autômato de pilha)

(a) Na gramática usamos a categoria sintátice de números inteiros $\mathbf{n} \in \text{Num}$.

$$c ::= \mathbf{c_1}\,\texttt{;}\,\mathbf{c_2} | \texttt{push } \mathbf{n} | \texttt{add} | \texttt{sub} | \texttt{mul}$$

(b) Nossa solução usa listas finitas σ. Como no caso da memôria, Σ denota o conjunto de todos estados possíveis, nessa caso o conjunto de todas listas finitas de números inteiros. Vamos usar uma operação estrita $\bullet : \mathbb{N} \times \Sigma_\perp \to \Sigma_\perp$, tal que $n \bullet \sigma$ é a concatenação de n com a lista σ.. A lista vazia é denotada por ϵ. A idéia é que uma lista represente uma pilha com o elemento mais esquerda ("a cabeça") sendo o topo da pilha. Observe que existem varias outras formalizações corretas, por exemplo um par de "apontador de pilha" e um memôria.

(c) A denotação de um comando é uma função $\Sigma_\perp \to \Sigma_\perp$ definido pelas equações semânticas

$$\begin{aligned}
S[\![\mathbf{c_1}\,\mathtt{;}\,\mathbf{c_2}]\!] &= S[\![\mathbf{c_2}]\!] \circ S[\![\mathbf{c_1}]\!] \\
S[\![\mathtt{push}\ \mathbf{n}]\!]\sigma &= A[\![\mathbf{n}]\!] \bullet \sigma \\
S[\![\mathtt{add}]\!](\mathbf{n_1} \bullet (\mathbf{n_2} \bullet \sigma)) &= (A[\![\mathbf{n_1}]\!] + A[\![\mathbf{n_2}]\!]) \bullet \sigma \\
S[\![\mathtt{add}]\!](\mathbf{n_1} \bullet \epsilon) &= \perp \\
S[\![\mathtt{add}]\!](\epsilon) &= \perp
\end{aligned}$$

As equações para `sub` e `mul` são analogas. A denotação dos números inteiros é simplesmente uma função $A[\![\cdot]\!] : \text{Num} \to \mathbb{Z}$

$$A[\![\mathtt{n}]\!] = n.$$