Fundamentals of Programming Languages VII

Program Semantics

Guoqiang Li

School of Software, Shanghai Jiao Tong University

Why Formal Semantics?

To understand how programs behave.

To build a mathematical model useful for program analysis and verification.

Three Kinds of Semantics

Operational semantics: describing the meaning of a programming language by specifying how it executes on an abstract machine.
Gordon Plotkin

Denotational semantics: defining the meaning of programming languages by mathematical concepts.
Christopher Strachey, Dana Scott

Axiomatic semantics: giving the meaning of a programming construct by axioms or proof rules in a program logic.
R.W. Floyd, C.A.R. Hoare

Three Kinds of Semantics

Operational semantics: very helpful in implementation

Denotational semantics: provides deep and widely applicable techniques for various languages

Axiomatic semantics: useful in developing and verifying programs

Three Kinds of Semantics

Different styles of semantics are dependent on each other. E.g.

Three Kinds of Semantics

Different styles of semantics are dependent on each other. E.g.

To show the proof rules of an axiomatic semantics are correct, use an underlying denotational or operational semantics.

To show an implementation correct wrt denotational semantics, need to show the operational and denotational semantics agree.

To justify an operational semantics, use a denotational semantics to abstract away from unimportant implementation details so to understand high-level computational behavior.

Basic Set Theory

Sets

$\{x \mid P(x)\}$: specify a set with property $P(x)$

Russell's paradox: $R = \{x \mid x \notin x\}$ is not a set.

So we assume all sets in this lecture are properly constructed.

$\emptyset$: the *null* or *empty* set

$\omega = \{0, 1, 2, ...\}$

Sets

Powerset: $\mathcal{P}ow(X) = \{Y \mid Y \subseteq X\}$.

Indexed set: $\{x_i \mid i \in I\}$.

Sets

Powerset: $\mathcal{P}ow(X) = \{Y \mid Y \subseteq X\}$.

Indexed set: $\{x_i \mid i \in I\}$.

Big union: Let X be a set of sets. $\bigcup X = \{a \mid \exists x \in X.a \in x\}$

When $X = \{x_i \mid i \in I\}$ for some indexing set I we write $\bigcup X$ as $\bigcup_{i \in I} x_i$.

Sets

Powerset: $\mathcal{P}ow(X) = \{Y \mid Y \subseteq X\}$.

Indexed set: $\{x_i \mid i \in I\}$.

Big union: Let X be a set of sets. $\bigcup X = \{a \mid \exists x \in X.a \in x\}$

When $X = \{x_i \mid i \in I\}$ for some indexing set I we write $\bigcup X$ as $\bigcup_{i \in I} x_i$.

Big intersection: Let X be a nonempty set of sets.
$\bigcap X = \{a \mid \forall x \in X.a \in x\}$

When $X = \{x_i \mid i \in I\}$ for a nonempty indexing set I we write $\bigcap X$ as $\bigcap_{i \in I} x_i$.

Sets

Product: $X \times Y = \{(a, b) \mid a \in X \ \& \ b \in Y\}$.

More generally, $X_1 \times X_2 \times ... \times X_n$ consists of the set of n-tuples $(x_1, x_2, ..., x_n) = (x_1, (x_2, (x_3, ...)))$.

Sets

Product: $X \times Y = \{(a, b) \mid a \in X \ \& \ b \in Y\}$.

More generally, $X_1 \times X_2 \times ... \times X_n$ consists of the set of n-tuples $(x_1, x_2, ..., x_n) = (x_1, (x_2, (x_3, ...)))$.

Disjoint union:
$X_0 \uplus X_1 \uplus \cdots \uplus X_n = (\{0\} \times X_0) \cup (\{1\} \times X_1) \cup \ldots \cup (\{n\} \times X_n)$

Set difference: $X \backslash Y = \{x \mid x \in X \ \& \ x \notin Y\}$

Sets

Product: $X \times Y = \{(a,b) \mid a \in X \ \& \ b \in Y\}$.

More generally, $X_1 \times X_2 \times ... \times X_n$ consists of the set of n-tuples $(x_1, x_2, ..., x_n) = (x_1, (x_2, (x_3, ...)))$.

Disjoint union:
$X_0 \uplus X_1 \uplus \cdots \uplus X_n = (\{0\} \times X_0) \cup (\{1\} \times X_1) \cup \ldots \cup (\{n\} \times X_n)$

Set difference: $X \backslash Y = \{x \mid x \in X \ \& \ x \notin Y\}$

The axiom of foundation: Any descending chain of memberships

$$\ldots b_n \in \ldots \in b_1 \in b_0$$

must be finite. Thus no set can be a member of itself. It is an assumption generally made in set theory.

Relations and Functions

A binary relation between X and Y is an element of $\mathcal{P}ow(X \times Y)$.

When R is a relation $R \subseteq X \times Y$, we write xRy for $(x, y) \in R$.

Relations and Functions

A binary relation between X and Y is an element of $\mathcal{P}ow(X \times Y)$.

When R is a relation $R \subseteq X \times Y$, we write xRy for $(x, y) \in R$.

A partial function from X to Y is a relation $f \subseteq X \times Y$ with

$$\forall x, y, y'.(x, y) \in f \ \& \ (x, y') \in f \Rightarrow y = y'$$

We write $f(x) = y$ when $(x, y) \in f$ for some y and say $f(x)$ is defined, otherwise $f(x)$ is undefined. Sometimes we write $f : x \mapsto y$ or $x \mapsto y$ when f is understood, for $y = f(x)$

Relations and Functions

A binary relation between X and Y is an element of $\mathcal{P}ow(X \times Y)$.

When R is a relation $R \subseteq X \times Y$, we write xRy for $(x, y) \in R$.

A partial function from X to Y is a relation $f \subseteq X \times Y$ with

$$\forall x, y, y'.(x, y) \in f \ \& \ (x, y') \in f \Rightarrow y = y'$$

We write $f(x) = y$ when $(x, y) \in f$ for some y and say $f(x)$ is defined, otherwise $f(x)$ is undefined. Sometimes we write $f : x \mapsto y$ or $x \mapsto y$ when f is understood, for $y = f(x)$

A (total) function from X to Y is a special partial function such that $\forall x \in X.\exists y \in Y.f(x) = y$.

Write $(X \rightharpoonup Y)$ for the set of all partial function from X to Y, and $(X \to Y)$ for the set of all total functions.

Relations and Functions

Lambda notation To write a function without naming it.
$\lambda x \in X.e = \{(x,e) \mid x \in X\}$

Let $R \subseteq X \times Y$ and $S \subseteq Y \times Z$ be two relations. Their composition is
$S \circ R =_{def} \{(x,z) \in X \times Z \mid \exists y \in Y.(x,y) \in R \ \& \ (y,z) \in S\}$

For functions $f : X \to Y$ and $g : Y \to Z$, their composition is the function $g \circ f : X \to Z$.

Each set X is associated with an identity function
$Id_X = \{(x,x) \mid x \in X\}$.

A function $f : X \to Y$ has an inverse $g : Y \to X$ iff $g(f(x)) = x$ for all $x \in X$ and $f(g(y)) = y$ for all $y \in Y$. Then X and Y are said to be in $1-1$ correspondence.

Relations and Functions

Let $R : X \times Y$ and $A \subseteq X$. The direct image of A under R
$RA = \{y \in Y \mid \exists x \in A.\ (x, y) \in R\}$

Let $B \subseteq Y$. The inverse image of B under R
$R^{-1}B = \{x \in X \mid \exists y \in B.\ (x, y) \in R\}$

If R is an equivalence relation on X, then the $(R-)$equivalence class of an element $x \in X$ is $\{x\}_R =_{def} \{y \in X \mid yRx\}$.

Let $R^0 = Id_X$, define $R^{n+1} = R \circ R^n$ for all $n \geq 0$. The transitive closure of R is $R^+ = \bigcup_{n \in \omega} R^{n+1}$. The reflexive, transitive closure of R is $R^* = Id_X \cup R^+ = \bigcup_{n \in \omega} R^n$.

Georg Cantor's Diagonal Argument

Theorem

Let X be any set, X and $\mathcal{P}ow(X)$ are never in $1-1$ correspondence.

Georg Cantor's Diagonal Argument

Theorem

Let X be any set, X and $\mathcal{P}ow(X)$ are never in $1-1$ correspondence.

Proof

Suppose there exists a 1-1 correspondence $\theta : X \to \mathcal{P}ow(X)$. Form the set $Y = \{x \in X \mid x \notin \theta(x)\}$. Now $Y \in \mathcal{P}ow(X)$ and is in correspondence with some $y \in X$, i.e. $\theta(y) = Y$.

Georg Cantor's Diagonal Argument

Theorem

Let X be any set, X and $\mathcal{P}ow(X)$ are never in $1-1$ correspondence.

Proof

Suppose there exists a 1-1 correspondence $\theta : X \to \mathcal{P}ow(X)$. Form the set $Y = \{x \in X \mid x \notin \theta(x)\}$. Now $Y \in \mathcal{P}ow(X)$ and is in correspondence with some $y \in X$, i.e. $\theta(y) = Y$.

- If $y \in Y$ then $y \notin \theta(y) = Y$.
- If $y \notin Y = \theta(y)$ then $y \in Y$.

So the correspondence θ does not exist at all.

Georg Cantor's Diagonal Argument

	$\theta(x_0)$	$\theta(x_1)$	$\theta(x_2)$	$\cdots$	$\theta(x_j)$	$\cdots$
x_0	0	1	1	$\cdots$	1	$\cdots$
x_1	1	1	1	$\cdots$	1	$\cdots$
x_2	0	0	0	$\cdots$	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	
x_i	0	1	0	$\cdots$	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	
x_i	0	1	0	$\cdots$	1	$\cdots$

In the ith row and jth column is placed 1 if $x_i \in \theta(x_j)$ and 0 otherwise.

Operational semantics

IMP- A Simple Imperative Language

Some syntactic sets in **IMP**.

- numbers **N**, consisting of all integer numbers, ranged over by metavariables n, m
- truth values **T**=$\{$**true**, **false**$\}$,
- locations **Loc**, ranged over by X, Y
- arithmetic expressions **Aexp**, ranged over by a
- boolean expressions **Bexp**, ranged over by b
- commands **Com**, ranged over by c

Sometimes we use metavariable which are primed or subscripted, e.g. X', X_0 for locations.

IMP- A Simple Imperative Language

The syntax of **IMP** defined by BNF (Backus-Naur form).

- For **Aexp**: $a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$
- For **Bexp**:
  $b ::= \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$
- For **Com**:
  $c ::= \textbf{skip} \mid X := a \mid c_0; c_1 \mid \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \mid \textbf{while } b \textbf{ do } c$

IMP- A Simple Imperative Language

The syntax of **IMP** defined by BNF (Backus-Naur form).

- For **Aexp**: $a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$
- For **Bexp**:
  $b ::= \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$
- For **Com**:
  $c ::= \textbf{skip} \mid X := a \mid c_0; c_1 \mid \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \mid \textbf{while } b \textbf{ do } c$
- From set-theoretic point of view, this notation gives an inductive definition of the syntactic sets, the least sets closed under the formation rules.
- Syntactic equivalence $\equiv$. e.g. $3 + 4 \not\equiv 4 + 3$.

Evaluation of Arithmetic Expressions

- The set of states consists of functions $\sigma : \mathbf{Loc} \to \mathbf{N}$.
- A configuration is a pair $\langle a, \sigma \rangle$, where a is an arithmetic expression and σ a state.
- An evaluation relation between pairs and numbers $\langle a, \sigma \rangle \to n$

Structural Operational Semantics

Evaluation of numbers $\langle n, \sigma \rangle \to n$

Evaluation of locations $\langle X, \sigma \rangle \to \sigma(X)$

Evaluation of sums

$$\frac{\langle a_0, \sigma \rangle \to n_0 \qquad \langle a_1, \sigma \rangle \to n_1 \qquad n \text{ is the sum of } n_0 \text{ and } n_1}{\langle a_0 + a_1, \sigma \rangle \to n}$$

Evaluation of subtractions

$$\frac{\langle a_0, \sigma \rangle \to n_0 \qquad \langle a_1, \sigma \rangle \to n_1 \qquad n \text{ is the result of subtracting } n_1 \text{ from } n_0}{\langle a_0 - a_1, \sigma \rangle \to n}$$

Evaluation of products

$$\frac{\langle a_0, \sigma \rangle \to n_0 \qquad \langle a_1, \sigma \rangle \to n_1 \qquad n \text{ is the product of } n_0 \text{ and } n_1}{\langle a_0 \times a_1, \sigma \rangle \to n}$$

Derivation Tree

$$\dfrac{\dfrac{\dfrac{}{\langle \mathit{Init}, \sigma_0 \rangle \to 0} \quad \dfrac{}{\langle 5, \sigma_0 \rangle \to 5}}{\langle (\mathit{Init} + 5), \sigma_0 \rangle \to 5} \quad \dfrac{\dfrac{}{\langle 7, \sigma_0 \rangle \to 7} \quad \dfrac{}{\langle 9, \sigma_0 \rangle \to 9}}{\langle 7 + 9, \sigma_0 \rangle \to 16}}{\langle (\mathit{Init} + 5) + (7 + 9), \sigma_0 \rangle \to 21}$$

Equivalence of Arithmetic Expressions

Two arithmetic expressions are equivalent if they evaluate to the same value in all states.

$a_0 \sim a_1$ iff $\forall \sigma \in \Sigma \, \forall n \in \mathbf{N}. \, \langle a_0, \sigma \rangle \to n \Leftrightarrow \langle a_1, \sigma \rangle \to n$

The Evaluation of Boolean Expressions

$$\langle \textbf{true}, \sigma \rangle \to \textbf{true} \qquad \langle \textbf{false}, \sigma \rangle \to \textbf{false}$$

$$\frac{\langle a_0, \sigma \rangle \to n \qquad \langle a_1, \sigma \rangle \to n}{\langle a_0 = a_1, \sigma \rangle \to \textbf{true}} \qquad \frac{\langle a_0, \sigma \rangle \to n \qquad \langle a_1, \sigma \rangle \to m \qquad n \not\equiv m}{\langle a_0 = a_1, \sigma \rangle \to \textbf{false}}$$

$$\frac{\langle a_0, \sigma \rangle \to n \qquad \langle a_1, \sigma \rangle \to m \qquad \text{if } n \text{ is less than or equal to } m}{\langle a_0 \leq a_1, \sigma \rangle \to \textbf{true}}$$

$$\frac{\langle a_0, \sigma \rangle \to n \qquad \langle a_1, \sigma \rangle \to m \qquad \text{if } n \text{ is not less than or equal to } m}{\langle a_0 \leq a_1, \sigma \rangle \to \textbf{false}}$$

$$\frac{\langle b, \sigma \rangle \to \textbf{true}}{\langle \neg b, \sigma \rangle \to \textbf{false}} \qquad \frac{\langle b, \sigma \rangle \to \textbf{false}}{\langle \neg b, \sigma \rangle \to \textbf{true}}$$

$$\frac{\langle b_0, \sigma \rangle \to t_0 \qquad \langle b_1, \sigma \rangle \to t_1 \qquad \text{if } t \text{ is } \textbf{true} \text{ iff } t_0 \equiv t_1 \equiv \textbf{true}}{\langle b_0 \wedge b_1, \sigma \rangle \to t}$$

$$\frac{\langle b_0, \sigma \rangle \to t_0 \qquad \langle b_1, \sigma \rangle \to t_1 \qquad \text{if } t \text{ is } \textbf{false} \text{ iff } t_0 \equiv t_1 \equiv \textbf{false}}{\langle b_0 \vee b_1, \sigma \rangle \to t}$$

The Execution of Commands

A (command) configuration is a pair $\langle c, \sigma \rangle$ where c is a command and σ a state. The execution of commands are defined via relations $\langle c, \sigma \rangle \to \sigma'$

Notation. Write $\sigma[m/X]$ for the state satisfying

$$\sigma[m/X](Y) = \begin{cases} m & \text{if } Y = X \\ \sigma(Y) & \text{if } Y \neq X \end{cases}$$

The Execution of Commands

Atomic commands

$$\langle \textbf{skip}, \sigma \rangle \to \sigma \qquad \frac{\langle a, \sigma \rangle \to m}{\langle X := a, \sigma \rangle \to \sigma[m/X]}$$

Sequencing

$$\frac{\langle c_0, \sigma \rangle \to \sigma'' \qquad \langle c_1, \sigma'' \rangle \to \sigma'}{\langle c_0; c_1, \sigma \rangle \to \sigma'}$$

Conditionals

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

While-loops

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

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

Big Step Semantics

To see the semantics just defined is a big step semantics, consider the following program:

$$
\begin{array}{lcl}
\mathit{Factorial} & \equiv & Y := 1; \\
 & & \textbf{while } X > 1 \textbf{ do} \\
 & & \quad \{Y := Y \times X;\ X := X - 1\}; \\
 & & Z := Y
\end{array}
$$

Let σ be a state with $\sigma(X) = 3$, what is the state σ' such that $\langle \mathit{Factorial}, \sigma \rangle \to \sigma'$? Construct the derivation tree.

Equivalence of Commands

Definition

$c_0 \sim c_1$ iff $\forall \sigma, \sigma' \in \Sigma.\ \langle c_0, \sigma \rangle \to \sigma' \Leftrightarrow \langle c_1, \sigma \rangle \to \sigma'$

Equivalence of Commands

Definition

$c_0 \sim c_1$ iff $\forall \sigma, \sigma' \in \Sigma.\ \langle c_0, \sigma \rangle \to \sigma' \Leftrightarrow \langle c_1, \sigma \rangle \to \sigma'$

Let $w \equiv$ **while** b **do** c with $b \in$ **Bexp** and $c \in$ **Com**. Then

$$w \ \sim \ \textbf{if } b \textbf{ then } c; w \textbf{ else skip}.$$

Equivalence of Commands

Definition

$c_0 \sim c_1$ iff $\forall \sigma, \sigma' \in \Sigma.\ \langle c_0, \sigma \rangle \to \sigma' \Leftrightarrow \langle c_1, \sigma \rangle \to \sigma'$

Let $w \equiv$ **while** b **do** c with $b \in$ **Bexp** and $c \in$ **Com**. Then

$$w \ \sim \ \textbf{if}\ b\ \textbf{then}\ c; w\ \textbf{else skip}.$$

Proof

Show that $\langle w, \sigma \rangle \to \sigma'$ iff $\langle$**if** b **then** $c; w$ **else skip**$, \sigma \rangle \to \sigma'$ for all states σ, σ'. Inspecting the rules with matching conclusions.

Small Step Semantics

For example,

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

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

$\langle n + m, \sigma\rangle \to_1 \langle p, \sigma\rangle$ $\quad$ p is the sume of n and m

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

Principles of Induction

Mathematical Induction

The principle of mathematical induction: Let $P(n)$ be a property of the natural number n. To show $P(n)$ holds for all natural numbers n it is sufficient to show

- $P(n)$ is true
- If $P(m)$ is true then so is $P(m+1)$ for any natural number m.

Mathematical Induction

The principle of mathematical induction: Let $P(n)$ be a property of the natural number n. To show $P(n)$ holds for all natural numbers n it is sufficient to show

- $P(n)$ is true
- If $P(m)$ is true then so is $P(m+1)$ for any natural number m.

i.e. $(P(0) \ \& \ (\forall m \in \omega.\, P(m) \Rightarrow P(m+1))) \Rightarrow \forall n \in \omega.\, P(n)$ where

- $P(0)$ is the induction basis
- $P(m)$ the induction hypothesis
- $(\forall m \in \omega.\, P(m) \Rightarrow P(m+1))$ the induction step.

Course-of-Values Induction

If a property Q's truth at $m+1$ depends on not just its truth at m but also its truth at other numbers preceding m as well, we strengthen the induction hypothesis to be $\forall k < m.\ Q(k)$. Then

- the basis: $\forall k < 0.\ Q(k)$: vacuously true.
- the induction step:
  $\forall m \in \omega.\ ((\forall k < m.\ Q(k)) \Rightarrow (\forall k < m+1.\ Q(k)))$ equivalent to
  $\forall m \in \omega.\ (\forall k < m.\ Q(k)) \Rightarrow Q(m)$

Course-of-Values Induction

If a property Q's truth at $m+1$ depends on not just its truth at m but also its truth at other numbers preceding m as well, we strengthen the induction hypothesis to be $\forall k < m.\ Q(k)$. Then

- the basis: $\forall k < 0.\ Q(k)$: vacuously true.
- the induction step:
  $\forall m \in \omega.\ ((\forall k < m.\ Q(k)) \Rightarrow (\forall k < m+1.\ Q(k)))$ equivalent to
  $\forall m \in \omega.\ (\forall k < m.\ Q(k)) \Rightarrow Q(m)$

So as a special form of mathematical induction is course-of-values induction: $(\forall m \in \omega.\ (\forall k < m.\ Q(k)) \Rightarrow Q(m)) \Rightarrow \forall n \in \omega.\ Q(n)$.

Structural Induction

Let $P(a)$ be a property of arithmetic expression a. To show $P(a)$ holds for all arithmetic expressions a it is sufficient to show:

- For all numerals m, $P(m)$ holds.
- For all locations X, $P(X)$ holds.
- For all arithmetic expressions a_0 and a_1, if $P(a_0)$ and $P(a_1)$ hold then so does $P(a_0 + a_1)$.
- Similarly with $P(a_0 - a_1)$ and $P(a_0 \times a_1)$.

Structural Induction: an Example

For all arithmetic expressions a, states σ and numbers m, m',

$$\langle a, \sigma \rangle \to m \ \ \& \ \ \langle a, \sigma \rangle \to m' \Rightarrow m = m'$$

Structural Induction: an Example

For all arithmetic expressions a, states σ and numbers m, m',

$$\langle a, \sigma \rangle \to m \ \ \& \ \ \langle a, \sigma \rangle \to m' \Rightarrow m = m'$$

Proof

By structural induction on arithmetic expressions a using induction hypothesis $P(a)$ where

$P(a)$ iff $\forall \sigma, m, m'.\ (\langle a, \sigma \rangle \to m \ \ \& \ \ \langle a, \sigma \rangle \to m' \Rightarrow m = m')$

- $a \equiv n$: since there is only one rule for evaluating $\langle n, \sigma \rangle$, trivial.
- $a \equiv a_0 + a_1$: Again one rule for evaluating $\langle a_0 + a_1, \sigma \rangle$. So

$$\langle a_0, \sigma \rangle \to m_0 \text{ and } \langle a_1, \sigma \rangle \to m_1 \text{ with } m = m_0 + m_1$$

$$\langle a_0, \sigma \rangle \to m'_0 \text{ and } \langle a_1, \sigma \rangle \to m'_1 \text{ with } m' = m'_0 + m'_1$$

By induction hypothesis applied to a_0, a_1 we obtain $m_0 = m'_0$ and $m_1 = m'_1$. Thus $m = m'$.

Structural Induction: an Example

For all arithmetic expressions a, states σ and numbers m, m',

$$\langle a, \sigma \rangle \to m \ \ \& \ \ \langle a, \sigma \rangle \to m' \Rightarrow m = m'$$

Proof

By structural induction on arithmetic expressions a using induction hypothesis $P(a)$ where

$P(a)$ iff $\forall \sigma, m, m'.\ (\langle a, \sigma \rangle \to m \ \ \& \ \ \langle a, \sigma \rangle \to m' \Rightarrow m = m')$

- $a \equiv n$: since there is only one rule for evaluating $\langle n, \sigma \rangle$, trivial.
- $a \equiv a_0 + a_1$: Again one rule for evaluating $\langle a_0 + a_1, \sigma \rangle$. So

  $$\langle a_0, \sigma \rangle \to m_0 \text{ and } \langle a_1, \sigma \rangle \to m_1 \text{ with } m = m_0 + m_1$$

  $$\langle a_0, \sigma \rangle \to m'_0 \text{ and } \langle a_1, \sigma \rangle \to m'_1 \text{ with } m' = m'_0 + m'_1$$

  By induction hypothesis applied to a_0, a_1 we obtain $m_0 = m'_0$ and $m_1 = m'_1$. Thus $m = m'$.
- The remaining cases are similar.

Well-Founded Relation

A well-founded relation is a binary relation $\prec$ on a set A such that there are no infinite descending chains

$$\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$$

. If $a \prec b$ then a is a predecessor of b.

Well-Founded Relation

Proposition

The relation $\prec$ on set A is well-founded iff any nonempty subset Q of A has a minimal element, i.e. an element m with

$$m \in Q \;\&\; \forall b \prec m . b \notin Q$$

Well-Founded Relation

Proposition

The relation $\prec$ on set A is well-founded iff any nonempty subset Q of A has a minimal element, i.e. an element m with

$$m \in Q \;\&\; \forall b \prec m.b \notin Q$$

Proof

($\Leftarrow$) Suppose every nonempty subset of A has a minimal element, but there is an infinite chain $\cdots \prec a_1 \prec a_0$. The set $\{a_i \mid i \in \omega\}$ would have no minimal element, a contradiction.

Well-Founded Relation

Proposition

The relation $\prec$ on set A is well-founded iff any nonempty subset Q of A has a minimal element, i.e. an element m with

$$m \in Q \ \& \ \forall b \prec m.b \notin Q$$

Proof

($\Leftarrow$) Suppose every nonempty subset of A has a minimal element, but there is an infinite chain $\cdots \prec a_1 \prec a_0$. The set $\{a_i \mid i \in \omega\}$ would have no minimal element, a contradiction.

($\Rightarrow$) Take any element a_0 from Q. Inductively, assume a chain $a_n \prec \cdots \prec a_0$ has been constructed inside Q. If there is $b \prec a_n$ with $b \in Q$, take $a_{n+1} = b$, otherwise stop the construction. As $\prec$ is well-founded, the chain is finite whose least element is minimal in Q.

Principle of Well-Founded Induction

Proposition

Let $\prec$ be well founded on set A, and P be a property. Then

$$\forall a.P(a) \text{ iff } \forall a \in A.((\forall b \prec a.\ P(b)) \Rightarrow P(a))$$

Principle of Well-Founded Induction

Proposition

Let $\prec$ be well founded on set A, and P be a property. Then

$$\forall a.P(a) \text{ iff } \forall a \in A.((\forall b \prec a.\ P(b)) \Rightarrow P(a))$$

Proof

($\Rightarrow$) Trivial.

Principle of Well-Founded Induction

Proposition

Let $\prec$ be well founded on set A, and P be a property. Then

$$\forall a.P(a) \text{ iff } \forall a \in A.((\forall b \prec a.\ P(b)) \Rightarrow P(a))$$

Proof

($\Rightarrow$) Trivial.

($\Leftarrow$) Suppose $\forall a \in A.((\forall b \prec a.\ P(b)) \Rightarrow P(a))$ but $\neg P(a)$ for some $a \in A$. The set $\{a \in A \mid \neg P(a)\}$ has a minimal element m. Then $\forall b \prec m.P(b)$ but $\neg P(m)$, contradicting the assumption.

Principle of Well-Founded Induction

Proposition

Let $\prec$ be well founded on set A, and P be a property. Then

$$\forall a.P(a) \text{ iff } \forall a \in A.((\forall b \prec a.\ P(b)) \Rightarrow P(a))$$

Proof

($\Rightarrow$) Trivial.

($\Leftarrow$) Suppose $\forall a \in A.((\forall b \prec a.\ P(b)) \Rightarrow P(a))$ but $\neg P(a)$ for some $a \in A$. The set $\{a \in A \mid \neg P(a)\}$ has a minimal element m. Then $\forall b \prec m.P(b)$ but $\neg P(m)$, contradicting the assumption.

In mathematics this principle is called Noetherian induction after the German algebraist Emmy Noether.

Principle of Well-Founded Induction

The proposition provides an alternative to proofs by well-founded induction.

To show property P holds for every element in a well-founded set A, it is sufficient to show the subset of counterexamples $\{a \in A \mid \neg P(a)\}$ is empty.

Suppose it is nonempty, there is a minimal element m contradicting the assumption $(\forall b \prec m.P(b)) \Rightarrow P(m)$.

An Example

Euclid's algorithm for the greatest common divisor of M, N.

$$\mathit{Euclid} \quad \equiv \quad \textbf{while } \neg(M = N) \textbf{ do}$$
$$\textbf{if } M \leq N \textbf{ then } N := N - M \textbf{ else } M := M - N$$

An Example

Theorem

For all states σ, $\sigma(M) \geq 1 \ \ \& \ \ \sigma(N) \geq 1 \Rightarrow \exists \sigma'.\ \langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$.

An Example

Theorem

For all states σ, $\sigma(M) \geq 1 \;\&\; \sigma(N) \geq 1 \Rightarrow \exists \sigma'.\, \langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$.

Proof

Let $S = \{\sigma \in \Sigma \mid \sigma(M) \geq 1 \;\&\; \sigma(N) \geq 1\}$ and $\prec$ by

$$\sigma' \prec \sigma \quad \text{iff} \quad (\sigma'(M) \leq \sigma(M) \;\&\; \sigma'(N) \leq \sigma(N)) \;\&\; \neg(\sigma'(M) = \sigma(M) \;\&\; \sigma'(N) = \sigma(N)).$$

An Example

Theorem

For all states σ, $\sigma(M) \geq 1 \ \ \& \ \ \sigma(N) \geq 1 \Rightarrow \exists \sigma'.\ \langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$.

Proof

Let $S = \{\sigma \in \Sigma \mid \sigma(M) \geq 1 \ \ \& \ \ \sigma(N) \geq 1\}$ and $\prec$ by

$$\sigma' \prec \sigma \quad \text{iff} \quad (\sigma'(M) \leq \sigma(M) \ \ \& \ \ \sigma'(N) \leq \sigma(N)) \ \ \& \\ \neg(\sigma'(M) = \sigma(M) \ \& \ \sigma'(N) = \sigma(N)).$$

Then $\prec$ is well-founded. Let $P(\sigma) = \exists \sigma'.\langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$. Suppose $\forall \sigma' \prec \sigma.P(\sigma')$, we show $P(\sigma)$ with two cases:

An Example

Theorem

For all states σ, $\sigma(M) \geq 1 \ \ \& \ \ \sigma(N) \geq 1 \Rightarrow \exists \sigma'.\ \langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$.

Proof

Let $S = \{\sigma \in \Sigma \mid \sigma(M) \geq 1 \ \ \& \ \ \sigma(N) \geq 1\}$ and $\prec$ by

$$\sigma' \prec \sigma \quad \text{iff} \quad (\sigma'(M) \leq \sigma(M) \ \ \& \ \ \sigma'(N) \leq \sigma(N)) \ \ \& \\ \neg(\sigma'(M) = \sigma(M) \ \& \ \sigma'(N) = \sigma(N)).$$

Then $\prec$ is well-founded. Let $P(\sigma) = \exists \sigma'.\langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$. Suppose $\forall \sigma' \prec \sigma.P(\sigma')$, we show $P(\sigma)$ with two cases:

I. $\sigma(M) = \sigma(N)$ and II. $\sigma(M) \neq \sigma(N)$

An Example

Theorem

For all states σ, $\sigma(M) \geq 1 \ \ \& \ \ \sigma(N) \geq 1 \Rightarrow \exists \sigma'.\ \langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$.

Proof

Let $S = \{\sigma \in \Sigma \mid \sigma(M) \geq 1 \ \ \& \ \ \sigma(N) \geq 1\}$ and $\prec$ by

$$\sigma' \prec \sigma \quad \text{iff} \quad (\sigma'(M) \leq \sigma(M) \ \ \& \ \ \sigma'(N) \leq \sigma(N)) \ \ \& \\ \neg(\sigma'(M) = \sigma(M) \ \& \ \sigma'(N) = \sigma(N)).$$

Then $\prec$ is well-founded. Let $P(\sigma) = \exists \sigma'.\langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$. Suppose $\forall \sigma' \prec \sigma.P(\sigma')$, we show $P(\sigma)$ with two cases:

I. $\sigma(M) = \sigma(N)$ and II. $\sigma(M) \neq \sigma(N)$

Argue in both cases that $\langle \mathit{Euclid}, \sigma \rangle \rightarrow \sigma'$ for some σ'. Then conclude $\forall \sigma \in S.P(\sigma)$ by well-founded induction.

Induction on Derivations

A rule instance is a pair X/y with premises X and conclusion y. Usually we write X/y as

$$\frac{}{y} \text{ if } X = \emptyset, \text{ and } \frac{x_1, \cdots, x_n}{y} \text{ if } X = \{x_1, \cdots, x_n\}$$

Let R be a set of rule instances. An R-derivation of y is either a rule instance $\emptyset/y$ or a pair $\{d_1, \cdots, d_n\}/y$ where $\{x_1, \cdots, x_n\}/y$ is a rule instance and d_i an R-derivation of x_i for all $1 \leq i \leq n$.

Write $d \Vdash_R y$ to mean d is an R-derivation of y.

Induction on Derivations

A rule instance is a pair X/y with premises X and conclusion y. Usually we write X/y as

$$\frac{}{y} \text{ if } X = \emptyset, \text{ and } \frac{x_1, \cdots, x_n}{y} \text{ if } X = \{x_1, \cdots, x_n\}$$

Let R be a set of rule instances. An R-derivation of y is either a rule instance $\emptyset/y$ or a pair $\{d_1, \cdots, d_n\}/y$ where $\{x_1, \cdots, x_n\}/y$ is a rule instance and d_i an R-derivation of x_i for all $1 \leq i \leq n$.

Write $d \Vdash_R y$ to mean d is an R-derivation of y.

- $(\emptyset/y) \Vdash_R y$ if $(\emptyset/y) \in R$
- $(\{d_1, \cdots, d_n\}/y) \Vdash_R y$ if $(\{x_1, \cdots, x_n\}/y) \in R$ and $d_1 \Vdash_R x_1, \ldots, d_n \Vdash_R x_n$

Induction on Derivations

A derivation d' is an immediate subderivation of d, written $d' \prec_1 d$, iff d has the form D/y with $d' \in D$.

Let $\prec$ be the transitive closure of $\prec_1$ ($\prec_1^+$). We say d' is a proper subderivation of d iff $d' \prec d$.

Since derivations are finite, both $\prec_1$ and $\prec$ are well-founded.

Induction on Derivations

Theorem

Let c be a command and σ_0 a state. If $\langle c, \sigma_0 \rangle \to \sigma$ and $\langle c, \sigma_0 \rangle \to \sigma'$, then $\sigma = \sigma_1$.

Proof

By well-founded induction on the proper subderivation relation $\prec$.

For any derivation d, let $P(d)$ be the following property

$$\forall c \in \mathbf{Com}, \sigma_0, \sigma, \sigma_1 \in \Sigma. d \Vdash \langle c, \sigma_0 \rangle \to \sigma \;\&\; \langle c, \sigma_0 \rangle \to \sigma_1 \Rightarrow \sigma = \sigma_1$$

Show that $\forall d' \prec d.P(d')$ implies $P(d)$ by inspecting the structure of c.

Induction on Derivations

Proposition

$$\forall c \in \mathbf{Com}, \sigma, \sigma' \in \Sigma.\ \langle \mathbf{while\ true\ do}\ c, \sigma \rangle \not\to \sigma'$$

Proof

Abbreviate $w \equiv \mathbf{while\ true\ do}\ c$. Suppose the set $\{d \mid \exists \sigma, \sigma' \in \Sigma.\ d \Vdash \langle w, \sigma \rangle \to \sigma'\}$ is nonempty. There is a minimal derivation d in the form

$$\dfrac{\dfrac{\vdots}{\langle \mathbf{true}, \sigma \rangle \to \mathbf{true}} \quad \dfrac{\vdots}{\langle c, \sigma \rangle \to \sigma''} \quad \dfrac{\vdots}{\langle w, \sigma'' \rangle \to \sigma'}}{\langle w, \sigma \rangle \to \sigma'}$$

But this contains a proper subderivation $d' \Vdash \langle w, \sigma'' \rangle \to \sigma'$, contradicting the minimality of d.

Definition by Induction

Definition by well-founded induction, also called well-founded recursion, e.g.

$$size(a) = \begin{cases} 1 & \text{if } a \equiv n \text{ or } X \\ 1 + size(a_0) + size(a_1) & \text{if } a = a_0 + a_1, \\ \vdots & \end{cases}$$

The Denotational Semantics of **IMP**

Motivation

Operational semantics is too concrete, built out of syntax, is hard to compare two programs written in different programming languages.

e.g.

$$c_0 \sim c_1 \text{ iff } (\forall \sigma, \sigma'.\ \langle c_0, \sigma \rangle \to \sigma') \Leftrightarrow \langle c_1, \sigma \rangle \to \sigma'$$

iff

$$\{(\sigma, \sigma') \mid \langle c_0, \sigma \rangle \to \sigma'\} = \{(\sigma, \sigma') \mid \langle c_1, \sigma \rangle \to \sigma'\}$$

i.e. c_0 and c_1 determine the same partial function on states.

So we take the denotation of a command to be a partial function on states.

Denotations of **Aexp**

Define the semantic function $\mathcal{A} : \mathbf{Aexp} \to (\Sigma \to \mathbf{N})$

$$
\begin{array}{rcl}
\mathcal{A}[\![n]\!] & = & \{(\sigma, n) \mid \sigma \in \Sigma\} \\
\mathcal{A}[\![X]\!] & = & \{(\sigma, \sigma(X)) \mid \sigma \in \Sigma\} \\
\mathcal{A}[\![a_0 + a_1]\!] & = & \{(\sigma, n_0 + n_1) \mid (\sigma, n_0) \in \mathcal{A}[\![a_0]\!] \ \& \ (\sigma, n_1) \in \mathcal{A}[\![a_1]\!]\} \\
\mathcal{A}[\![a_0 - a_1]\!] & = & \{(\sigma, n_0 - n_1) \mid (\sigma, n_0) \in \mathcal{A}[\![a_0]\!] \ \& \ (\sigma, n_1) \in \mathcal{A}[\![a_1]\!]\} \\
\mathcal{A}[\![a_0 \times a_1]\!] & = & \{(\sigma, n_0 \times n_1) \mid (\sigma, n_0) \in \mathcal{A}[\![a_0]\!] \ \& \ (\sigma, n_1) \in \mathcal{A}[\![a_1]\!]\}
\end{array}
$$

The “$+$" on the left-hand side represents syntactic sign in **IMP** whereas the sign on the right represents sum on numbers. Similarly for “$-$", “$\times$".

Denotations of **Aexp**

The denotation of arithmetic expressions are actually total functions. Using λ-notation,

$$
\begin{array}{rcl}
\mathcal{A}[\![n]\!] & = & \lambda\sigma \in \Sigma.\, n \\
\mathcal{A}[\![X]\!] & = & \lambda\sigma \in \Sigma.\, \sigma(X) \\
\mathcal{A}[\![a_0 + a_1]\!] & = & \lambda\sigma \in \Sigma.\, (\mathcal{A}[\![a_0]\!]\sigma + \mathcal{A}[\![a_1]\!]\sigma) \\
\mathcal{A}[\![a_0 - a_1]\!] & = & \lambda\sigma \in \Sigma.\, (\mathcal{A}[\![a_0]\!]\sigma - \mathcal{A}[\![a_1]\!]\sigma) \\
\mathcal{A}[\![a_0 \times a_1]\!] & = & \lambda\sigma \in \Sigma.\, (\mathcal{A}[\![a_0]\!]\sigma \times \mathcal{A}[\![a_1]\!]\sigma)
\end{array}
$$

Denotations of **Bexp**

Define the semantic function $\mathcal{B} : \mathbf{Bexp} \to (\Sigma \to \mathbf{T})$

$$
\begin{array}{rcl}
\mathcal{B}[\![\mathbf{true}]\!] & = & \{(\sigma, \mathbf{true}) \mid \sigma \in \Sigma\} \\
\mathcal{B}[\![\mathbf{false}]\!] & = & \{(\sigma, \mathbf{false}) \mid \sigma \in \Sigma\} \\
\mathcal{B}[\![a_0 = a_1]\!] & = & \{(\sigma, \mathbf{true}) \mid \sigma \in \Sigma \ \&\ \mathcal{A}[\![a_0]\!]\sigma = \mathcal{A}[\![a_1]\!]\sigma\} \cup \\
 & & \{(\sigma, \mathbf{false}) \mid \sigma \in \Sigma \ \&\ \mathcal{A}[\![a_0]\!]\sigma \neq \mathcal{A}[\![a_1]\!]\sigma\} \cup \\
\mathcal{B}[\![\neg b]\!] & = & \{(\sigma, \neg_T t) \mid \sigma \in \Sigma \ \&\ (\sigma, t) \in \mathcal{B}[\![b]\!]\} \\
\mathcal{B}[\![b_0 \wedge b_1]\!] & = & \{(\sigma, t_0 \wedge_T t_1) \mid \sigma \in \Sigma \ \&\ (\sigma, t_0) \in \mathcal{B}[\![b_0]\!] \ \& \\
 & & (\sigma, t_1) \in \mathcal{B}[\![b_1]\!]\} \\
 & \ldots &
\end{array}
$$

The sign "$\wedge_T$" is the conjunction operation on truth values.

Denotations of **Com**

Define the compositional semantic function $\mathcal{C} : \mathbf{Aexp} \to (\Sigma \rightharpoonup \Sigma)$

$$
\begin{array}{rcl}
\mathcal{C}[\![\mathbf{skip}]\!] & = & \{(\sigma,\sigma) \mid \sigma \in \Sigma\} \\
\mathcal{C}[\![X := a]\!] & = & \{(\sigma,\sigma[n/X]) \mid \sigma \in \Sigma \ \&\ n = \mathcal{A}[\![a]\!]\sigma\} \\
\mathcal{C}[\![c_0; c_1]\!] & = & \mathcal{C}[\![c_1]\!] \circ \mathcal{C}[\![c_0]\!] \\
\mathcal{C}[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1]\!] & = & \{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{true} \ \&\ (\sigma,\sigma') \in \mathcal{C}[\![c_0]\!]\} \\
 & & \cup\{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{false} \ \&\ (\sigma,\sigma') \in \mathcal{C}[\![c_1]\!]\} \\
\mathcal{C}[\![\mathbf{while}\ b\ \mathbf{do}\ c]\!] & = & \mathit{fix}(\Gamma)
\end{array}
$$

where

$$
\begin{array}{rcl}
\Gamma(\varphi) = & \{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{true} \ \&\ (\sigma,\sigma') \in \varphi \circ \mathcal{C}[\![c]\!]\} \cup \\
 & \{(\sigma,\sigma) \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{false}\}
\end{array}
$$

Denotation of **while** -Loops

Let $w \equiv$ **while** b **do** c. Inspired by the equivalence $w \sim$ **if** b **then** $c; w$ **else skip**. We should have

$$\begin{aligned} \mathcal{C}[\![w]\!] = \quad & \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \textbf{true} \ \ \& \ \ (\sigma, \sigma') \in \mathcal{C}[\![c; w]\!]\} \cup \\ & \{(\sigma, \sigma) \mid \mathcal{B}[\![b]\!]\sigma = \textbf{false}\} \end{aligned}$$

Equivalence of the Semantics

Lemma

For all $a \in \mathbf{Aexp}$, $\mathcal{A}[\![a]\!] = \{(\sigma, n) \mid \langle a, \sigma \rangle \to n\}$.

Equivalence of the Semantics

Lemma

For all $a \in \mathbf{Aexp}$, $\mathcal{A}[\![a]\!] = \{(\sigma, n) \mid \langle a, \sigma \rangle \to n\}$.

Proof

Define the property P by $P(a) =_{def} \mathcal{A}[\![a]\!] = \{(\sigma, n) \mid \langle a, \sigma \rangle \to n\}$ and proceed by structural induction on arithmetic expressions.

Equivalence of the Semantics

Lemma

For all $a \in \mathbf{Aexp}$, $\mathcal{A}[\![a]\!] = \{(\sigma, n) \mid \langle a, \sigma \rangle \to n\}$.

Proof

Define the property P by $P(a) =_{def} \mathcal{A}[\![a]\!] = \{(\sigma, n) \mid \langle a, \sigma \rangle \to n\}$ and proceed by structural induction on arithmetic expressions.

Lemma

For all $b \in \mathbf{Bexp}$, $\mathcal{B}[\![b]\!] = \{(\sigma, t) \mid \langle b, \sigma \rangle \to t\}$.

Equivalence of the Semantics

Lemma

For all commands c and states σ, σ', $\langle c, \sigma \rangle \to \sigma' \Rightarrow (\sigma, \sigma') \in \mathcal{C}[\![c]\!]$.

Equivalence of the Semantics

Lemma

For all commands c and states σ, σ', $\langle c, \sigma \rangle \to \sigma' \Rightarrow (\sigma, \sigma') \in \mathcal{C}[\![c]\!]$.

Proof

Let $P(c, \sigma, \sigma') =_{def} (\sigma, \sigma') \in \mathcal{C}[\![c]\!]$. Use rule induction for commands.

Equivalence of the Semantics

Theorem

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

The Axiomatic Semantics of **IMP**

The Idea

Assertions in programs.

$$S := 0; N := 1$$
$$\{S = 0 \;\&\; N = 1\}$$
$$\textbf{while } \neg(N = 101) \textbf{ do } S := S + N; N := N + 1$$
$$\{S = \textstyle\sum_{1 \leq m \leq 100} m\}$$

Partial Correctness

Let A, B be assertions like those in **Bexp**, and c a command. We write

$$\{A\}c\{B\}$$

to mean: for all states σ which satisfy A (precondition) if the execution c from state σ terminates in state σ' then σ' satisfies B (postcondition).

Partial Correctness

Let A, B be assertions like those in **Bexp**, and c a command. We write

$$\{A\}c\{B\}$$

to mean: for all states σ which satisfy A (precondition) if the execution c from state σ terminates in state σ' then σ' satisfies B (postcondition).

$$\{\mathbf{true}\}\mathbf{while\ true\ do\ skip}\{\mathbf{false}\}$$

Partial Correctness

Let A, B be assertions like those in **Bexp**, and c a command. We write

$$\{A\}c\{B\}$$

to mean: for all states σ which satisfy A (precondition) if the execution c from state σ terminates in state σ' then σ' satisfies B (postcondition).

$$\{\mathbf{true}\}\mathbf{while\ true\ do\ skip}\{\mathbf{false}\}$$

In contrast to total correctness assertions $[A]c[B]$ — the execution of c from any state which satisfies A will terminate in a state which satisfies B.

Partial Correctness

Consider $\mathcal{C}[\![c]\!]$ as a total function in $(\Sigma \to \Sigma_\perp)$ instead of partial function in $(\Sigma \rightharpoonup \Sigma)$.

Partial Correctness

Consider $\mathcal{C}[\![c]\!]$ as a total function in $(\Sigma \to \Sigma_\perp)$ instead of partial function in $(\Sigma \rightharpoonup \Sigma)$.

Write $\sigma \models A$ to mean the state σ satisfies assertion A. Let $\perp \models A$ for any A. Then the meaning of $\{A\}c\{B\}$ will be

$$\forall \sigma \in \Sigma.\ \sigma \models A \Rightarrow \mathcal{C}[\![c]\!]\sigma \models B.$$

The Assertion Language **Assn**

Let i range over integer variables, **Intvar**. Extending **Aexp** with integer variables to be **Aexpv**:

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

Extending **Bexp** to be **Assn**:

$$\begin{aligned} A ::= \quad & \mathbf{true} \mid \mathbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid A_0 \wedge A_1 \mid \\ & A_0 \vee A_1 \mid \neg A \mid A_0 \Rightarrow A_1 \mid \forall i.A \mid \exists i.A \end{aligned}$$

Free Integer Variables

Define free integer variables in **Aexpv** or **Assn** expressions by structural induction.

$$FV(n) = FV(X) = \emptyset$$
$$FV(i) = \{i\}$$
$$FV(a_0 + a_1) = FV(a_0 - a_1) = FV(a_0 \times a_1) = FV(a_0) \cup FV(a_1)$$

$$FV(\mathbf{true}) = FV(\mathbf{false}) = \emptyset$$
$$FV(a_0 = a_1) = FV(a_0 \leq a_1) = FV(a_0) \cup FV(a_1)$$
$$FV(A_0 \wedge A_1) = FV(A_0 \vee A_1) = FV(A_0 \Rightarrow A_1) = FV(A_0) \cup FV(A_1)$$
$$FV(\neg A) = FV(A)$$
$$FV(\forall i.A) = FV(\exists i.A) = FV(A) \backslash \{i\}$$

Substitution

Define substitution for **Aexpv** or **Assn** expressions by structural induction.

$$n[a/i] \equiv n \qquad X[a/i] \equiv X$$
$$j[a/i] \equiv j \qquad i[a/i] \equiv a$$
$$(a_0 + a_1)[a/i] \equiv (a_0[a/i] + a_1[a/i])$$
$$\ldots$$
$$\mathbf{true}[a/i] \equiv \mathbf{true} \qquad \mathbf{false}[a/i] \equiv \mathbf{false}$$
$$(a_0 = a_1)[a/i] \equiv (a_0[a/i] = a_1[a/i])$$
$$(A_0 \wedge A_1)[a/i] \equiv (A_0[a/i] \wedge A_1[a/i])$$
$$(\neg A)[a/i] \equiv \neg(A[a/i])$$
$$(\forall j.A)[a/i] \equiv \forall j.(A[a/i]) \qquad (\forall i.A)[a/i] \equiv \forall i.A$$
$$(\exists j.A)[a/i] \equiv \exists j.(A[a/i]) \qquad (\exists i.A)[a/i] \equiv \exists i.A$$

The Meaning of Expressions, **Aexpv**

An interpretation is a function I : **Intvar** $\rightarrow$ **N** assigning an integer to each integer variable. The value of an expression $a \in$ **Aexpv** in an interpretation I and state σ is written $\mathcal{A}v[\![a]\!]I\sigma$ or $(\mathcal{A}v[\![a]\!](I))(\sigma)$.

$$
\begin{aligned}
&\mathcal{A}v[\![n]\!]I\sigma = n \\
&\mathcal{A}v[\![X]\!]I\sigma = \sigma(X) \\
&\mathcal{A}v[\![i]\!]I\sigma = I(i) \\
&\mathcal{A}v[\![a_0 + a_1]\!]I\sigma = \mathcal{A}v[\![a_0]\!]I\sigma + \mathcal{A}v[\![a_1]\!]I\sigma \\
&\mathcal{A}v[\![a_0 - a_1]\!]I\sigma = \mathcal{A}v[\![a_0]\!]I\sigma - \mathcal{A}v[\![a_1]\!]I\sigma \\
&\mathcal{A}v[\![a_0 \times a_1]\!]I\sigma = \mathcal{A}v[\![a_0]\!]I\sigma \times \mathcal{A}v[\![a_1]\!]I\sigma
\end{aligned}
$$

The Meaning of Assertions, **Assn**

Write $I[n/i]$ for the interpretation given by $I[n/i](j) = n$ if $j \equiv i$, and $I(j)$ otherwise.
For $A \in \mathbf{Assn}$, write $\sigma \models^I A$ to mean σ satisfies A in interpretation I.

$$
\begin{aligned}
&\sigma \models^I \mathbf{true} \\
&\sigma \models^I (a_0 = a_1) \text{ if } \mathcal{A}v[\![a_0]\!]I\sigma = \mathcal{A}v[\![a_1]\!]I\sigma \\
&\sigma \models^I A \wedge B \text{ if } \sigma \models^I A \text{ and } \sigma \models^I B \\
&\sigma \models^I A \Rightarrow B \text{ if } \sigma \not\models^I A \text{ or } \sigma \models^I B \\
&\sigma \models^I \forall i.A \text{ if } \sigma \models^{I[n/i]} A \text{ for all } n \in \mathbf{N} \\
&\sigma \models^I \exists i.A \text{ if } \sigma \models^{I[n/i]} A \text{ for some } n \in \mathbf{N} \\
&\bot \models^I A \\
&\ldots
\end{aligned}
$$

Partial Correctness Assertions

Write $A^I = \{\sigma \in \Sigma_\perp \mid \sigma \models^I A\}$.

- $\sigma \models^I \{A\}c\{B\}$ iff $(\sigma \models^I A \Rightarrow C[\![c]\!]\sigma \models^I B)$.
- $\models^I \{A\}c\{B\}$ iff $\forall \sigma \in \Sigma_\perp .\ \sigma \models^I \{A\}c\{B\}$
- Validity: $\models \{A\}c\{B\}$ iff
  $\sigma \models^I \{A\}c\{B\}$ for all interpretations I and states σ
- Similarly, A is valid, $\models A$, means $\sigma \models^I A$ for all interpretations I and states σ.

Proof Rules for Partial Correctness

The proof rules are called Hoare rules and the proof system Hoare logic.

$$\{A\}\ \textbf{skip}\ \{A\}$$

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

$$\frac{\{A\}c_0\{C\} \quad \{C\}c_1\{B\}}{\{A\}\ c_0; c_1\ \{B\}}$$

$$\frac{\{A \wedge b\}c_0\{B\} \quad \{A \wedge \neg b\}c_1\{B\}}{\{A\}\ \textbf{if}\ b\ \textbf{then}\ c_0\ \textbf{else}\ c_1\ \{B\}}$$

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

$$\frac{\models (A \Rightarrow A') \quad \{A'\}c\{B'\} \quad \models (B' \Rightarrow B)}{\{A\}\ c\ \{B\}}$$

Soundness of the Proof System

A rule is sound in the sense that if the rule's premise is valid then so is its conclusion. The proof system is sound if every rule is sound. Then by rule induction, every theorem obtained from the proof system is a valid partial correctness assertion.

Soundness of the Proof System

A rule is sound in the sense that if the rule's premise is valid then so is its conclusion. The proof system is sound if every rule is sound. Then by rule induction, every theorem obtained from the proof system is a valid partial correctness assertion.

Lemma

Let I be an interpretation, σ a state, and $X \in \mathbf{Loc}$.

- Let $a, a_0 \in \mathbf{Aexpv}$. Then

$$\mathcal{A}v[\![a_0[a/X]]\!]I\sigma = \mathcal{A}v[\![a_0]\!]I\sigma[\mathcal{A}v[\![a]\!]I\sigma/X]$$

- Let $B \in \mathbf{Assn}$. Then

$$\sigma \models^I B[a/X] \text{ iff } \sigma[\mathcal{A}[\![a]\!]\sigma/X] \models^I B$$

Soundness of the Proof System

A rule is sound in the sense that if the rule's premise is valid then so is its conclusion. The proof system is sound if every rule is sound. Then by rule induction, every theorem obtained from the proof system is a valid partial correctness assertion.

Lemma

Let I be an interpretation, σ a state, and $X \in \mathbf{Loc}$.

- Let $a, a_0 \in \mathbf{Aexpv}$. Then

$$\mathcal{A}v[\![a_0[a/X]]\!]I\sigma = \mathcal{A}v[\![a_0]\!]I\sigma[\mathcal{A}v[\![a]\!]I\sigma/X]$$

- Let $B \in \mathbf{Assn}$. Then

$$\sigma \models^I B[a/X] \text{ iff } \sigma[\mathcal{A}[\![a]\!]\sigma/X] \models^I B$$

Proof

By structural induction on a_0 and B respectively.

Soundness of the Proof System

Theorem

Let $\{A\}c\{B\}$ be a partial correctness assertion. If $\vdash \{A\}c\{B\}$ then $\models \{A\}c\{B\}$.

Using the Hoare Rules

Let $w \equiv (\textbf{while}\ X > 0\ \textbf{do}\ Y := X \times Y; X := X - 1)$, and show

$$\{X = n\ \&\ n \geq 0\ \&\ Y = 1\} w \{Y = n!\}$$

Take $I \equiv (Y \times X! = n!\ \&\ X \geq 0)$, then

$$\{I \wedge X > 0\} Y := X \times Y; X := X - 1 \{I\}$$

and so $\{I\} w \{I \wedge X \not> 0\}$.

Note $X = n\ \&\ n \geq 0\ \&\ Y = 1 \Rightarrow I$ and $I \wedge X \not> 0 \Rightarrow Y = n!$

Report

Rep8. Semantics and Key Properties of CCS. (Maximal 3 Students)

Rep9. Semantics and Key Properties of CSP. (Maximal 3 Students)