Iterative problem solving: The mapcode approach

Venkatesh Choppella

2024-02-27 12:11:49+05:30

IIIT Hyderabad

Contents

What are computers good at?

Computers are good at **repeatedly** doing a task.

1. They are fast.
2. They don't get tired.
3. They don't get bored.

Repeatedly doing a task is called **iteration**.

Programming: Instructing a computer what to do

Computers are used to solve problems that take an instance and return an answer after iterating on a task.

But they need to be **instructed**:

1. Where to start
2. What to do
3. When to stop
4. How to report the answer

Anatomy of a computation: computing 3!

instance

answer

$$
\begin{tikzcd}
3 \arrow[d, "\rho"] & & & & 6 \\
(3,1) \arrow[r, "F"] & (2,3) \arrow[r, "F"] & (1,6) \arrow[r, "F"] & (0,6) \arrow[r, "F"] & (0,6) \arrow[u, "\pi"']
\end{tikzcd}
$$

fixed point

state

initial state

1. Where to start: ρ
2. What to do: F
3. When to stop: fixed point
4. How to report answer: π.

The structure of states and maps

$$\begin{tikzcd}
3 \arrow[d, "\rho"] & & & & 6 \\
(3,1) \arrow[r, "F"] & (2,\ 3) \arrow[r, "F"] & (1,6) \arrow[r, "F"] & (0,6) \arrow[r, "F"] & (0,6) \arrow[u, "\pi"']
\end{tikzcd}$$

- accumulator
- counter

- ρ: maps instances to states
- F: maps states to states
- π: maps states to answers

Multiplication using addition and decrement

$$\begin{tikzcd}
(3,4) \arrow[d, "\rho"] & & & 12 \\
(3,4,0) \arrow[r, "F"] & (2,4,4) \arrow[r, "F"] & (1,4,8) \arrow[r, "F"] & (0,4,12) \arrow[u, "\pi"']
\end{tikzcd}$$

What we plan to do in these slides

Our goal in these slides is to

1. Introduce a simple mathematical theory of iteration
2. Define iterative problem solving
3. Implement iterative problem solving in Python

Contents

Definition 1 (Discrete Flow)
A discrete flow D is a pair

$$\langle X, F : X \to X \rangle$$

where

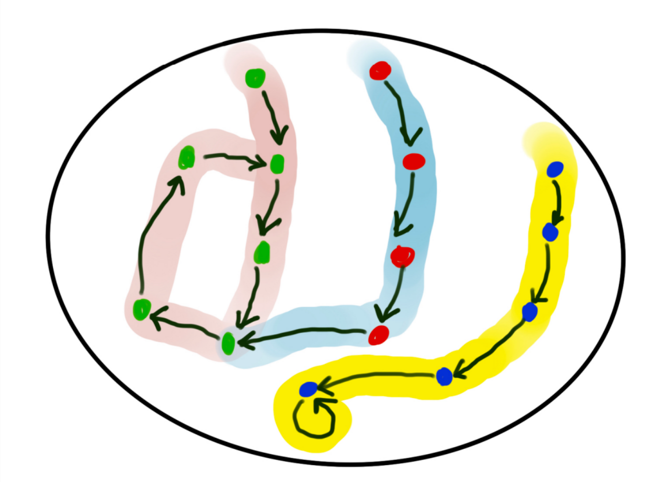
- X is a set called the **state space** of D.
- F is a function called the **dynamical map** of D.

Prime Notation

x' denotes the 'next' state.

$$\boxed{x' = F(x)}$$

Picture of a Discrete Flow



Exercise break: Examples & non-examples of Discrete Flow

Which of the following are discrete flows?

1. $X = \mathbb{N}$, $F_{\mathsf{inc}} = x \mapsto x + 1$
2. $X = \mathbb{N}$, $F_{\mathsf{sqr}} = x \mapsto x^2$
3. $X = \mathbb{R}$, $F_{\mathsf{cos}} = x \mapsto cosine(x)$
4. $X = \mathbb{R}$, $F_{?} = x \mapsto (x - 1)/x$
5. $X = \mathbb{N}$, $F_{countdown} = n \mapsto$ $\begin{cases} 0 & \text{if } n = 0 \\ n - 1 & \text{otherwise} \end{cases}$

Definition 2 (F-closed sets)
Let $D = (X, F : X \to X)$ be a discrete flow. A set S of X is F-**closed** if $F(S) \subseteq S$, i.e., for each $x \in S$, $F(x) \in S$.

Examples of closed subsets

Let $D = (X, F : X \to X)$ be a discrete flow. The following subsets of X are closed:

1. X
2. $\emptyset$

Definition 3 (Trajectory, Orbit)
Let $\langle X, F : X \to X \rangle$ be a discrete flow.

- The **trajectory** of an element $x \in X$ is the sequence

$$x, F(x), F^2(x), F^3(x), \ldots$$

- The **orbit** of x is the set

$$\{x, F(x), F^2(x), F^3(x), \ldots\}$$

More examples of F-closed subsets

1. $orb(x)$ where $x \in X$

2. $orb(S) = \cup_{x \in S}\, orb(x)$ where S is any subset of X

Definition 4 (Subflow)
Let $D = (X, F : X \to X)$ be a discrete flow. Let S be a subset X that is F-closed.

Then $D_S = (S, F|_S)$ is a discrete flow. [1] D_S is called a **subflow of** D.

[1] If $S \subseteq A$ and $F : A \to B$, then $F|_S : S \to B$ is the restriction of F to S.

The subflow generated by an element

Definition 5 (Generated Subflow)
Let $(X, F : X \to X)$ be a flow. Let $x \in X$. Then $(orb(x), F|_{orb(x)})$ is a subflow **generated** by x.

If $S \subseteq X$, then $(orb(S), F|_{orb(S)})$ is the subflow **generated** by S.

Definition 6 (Fixed Point)
Let $D = \langle X, F : X \to X \rangle$ be a discrete flow.

- $x \in X$ is a **fixed point** of F if $x = F(x)$.
- $\mathbf{fix}(\mathbf{F})$: the set of fixed points of F.

Exercise: Guess Fixed Points

1. $(\mathbb{N}, F_{\mathsf{inc}})$:

2. $(\mathbb{N}, F_{\mathsf{sqr}})$:

3. $(\mathbb{R}, F_{\mathsf{cos}})$: hint

4. $(\mathbb{N}, F_{countdown})$:

More examples of closed subsets

1. $\{x\}$, where x is a fixed point of F

2. $\mathit{fix}(F)$

Transient Points

Definition 7 (Transient Point)
Let $D = \langle X, F : X \to X \rangle$ be a discrete flow.

- $x \in X$ is **transient** if $x \neq F(x)$.

Reachability

Definition 8 (Reaches)
Let $(X, F : X \to X)$ be a discrete flow.

Let x and y be states in X.

x **reaches** y, alternatively y is **reachable from** x, if, for some $i \in \mathbb{N}$,

$$y = F^i(x)$$

Definition 9 (Convergent point)
Let $D = \langle X, F : X \to X \rangle$ be a discrete flow.

- $x \in X$ is a **(F-) convergent point** of F in X if it reaches a fixed point.
- $S \subseteq X$ is **convergent** if for each $x \in S$, x is convergent.
- $\mathbf{con}(\mathbf{F})$: the set of convergent points of F in X.

Exercise: Guess Convergent Points

1. $(\mathbb{N}, F_{\mathsf{inc}})$:

2. $(\mathbb{N}, F_{\mathsf{sqr}})$:

3. $(\mathbb{R}, F_{\mathsf{cos}})$: hint

4. $(\mathbb{N}, F_{countdown})$:

Fixed Point iteration in Python

```python
# Assumes x in con(F)
# returns element in fix(F)
# xprime ensures F(x) computed
# only once per iteration.
def loop(x,F):
    while True:
        xprime = F(x)
        if x == xprime:  # fixed point!
            break
        else:
            x = xprime
    return x
```

Limit Map

Let $D = \langle X, F : X \to X \rangle$ be a discrete flow.

The **limit map** of F, *F-infinity*, is the function

$$F^\infty : con(F) \to fix(F)$$

Limit Map in Python

```
# Limit Map
# takes a function F
# returns a function F_inf.
# F_inf takes an x in con(F)
# and returns an element in fix(F).
def limit_map(F):
    def F_inf(x): # assume: x in con(F)
        return loop(x,F)
    return F_inf
```

Contents

What is Iterative Problem Solving?

Definition 10 (Iterative Problem Solving)
An instance $\mathcal{A}$ of computational problem solving is a pair $\mathcal{A} = \langle \mathcal{P}, \mathcal{M} \rangle$ consisting of

1. A **problem** specification $\mathcal{P}$
2. A **mapcode machine** specification $\mathcal{M}$

In what follows, we present the Mapcode Approach to Iterative Problem solving[**?**].

Contents

IPS: Problem specification

A problem specification $\mathcal{P} = \langle \mathcal{F}, \mathcal{R}, \mathcal{N} \rangle$ consists of

- **Functional Requirement Specification** $\mathcal{F}$: What is the function that needs to be computed?
- **Resource Specification** $\mathcal{R}$: What are the primitive datatypes and operations on those datatypes available to compute the function?
- **Non-functional Requirements Specification** $\mathcal{N}$: What are the constraints on time, memory space, total cost, security, performance, and usability of the computational solution?

Functional Specification of a problem

A functional specification $\mathcal{F} = \langle I, A, f \rangle$ consists of the following:

- **Input Space** I : The set of all possible inputs.
- **Answer Space** A : The set from which answers are drawn.
- **Specification map** $f : I \to A$, the function to be computed.

Existence and Uniqueness of Answer for each Problem input

When constructing the specification map, one needs to be convinced that each problem input is indeed associated with a unique answer.

- **Existence**: For each problem input $p : I$, there is indeed a answer $a : A$.

- **Uniqueness**: If a_1 and a_2 are answers associated with a problem input p, then $a_1 = a_2$.

Example of a functional specification: GCD

Compute the Greatest Common Divisor of two natural numbers.

- Input Space: $\mathbb{N} \times \mathbb{N}$
- Answer Space: $\mathbb{N}$
- Specification map: $gcd : \mathbb{N}^2 \times \mathbb{N}$. For each pair of naturals (a, b), $gcd(a, b)$ is the largest number that divides both a and b.

Existence and Uniqueness for GCD

Let a, b be two naturals. Then

- **Existence**: The gcd is an element of all the common factors of a and b. This set is not empty, since clearly, 1 is a common factor for both a and b.

- **Uniqueness**: Let r_1 and r_2 be two gcd's of a and b. Then, since r_1 is the *greatest* common factor, $r_1 \geq r_2$. By a similar argument, $r_2 \geq r_1$. Hence $r_1 = r_2$.

Exercise: Define Functional Specifications

1. Factorial
2. The maximum element in a list
3. Searching a given element in a list
4. Reversing a list
5. Sorting a list

Exercise solution: Functional Specification of Factorial

1. Input space: $I = \mathbb{N}$ (natural numbers)
2. Answer space: $A = \mathbb{N}$ (natural numbers)
3. Specification map $f : I \to A$
4. $f(n) = \begin{cases} 1 & \text{if } n = 0 \\ \prod_{i=1}^{n} i & \text{if } n > 0 \end{cases}$

Clearly f is a function. For $n = 0$, the answer is unique. For $n > 0$, the answer is unique because $\prod$ is a total function.

Exercise solution: Functional Specification for List Search

1. Input space: $I = a : \alpha \times s : \mathsf{List}[\alpha]$ (List of elements of type α).

2. Answer space: $A = \{\mathsf{absent}\} \cup \{i : [0 \ \ .. \ \ |s| {-} 1]\}$.

3. Specification map $f : I \to A$

4. $f(s) = \begin{cases} \text{absent} & \text{if } a \notin \underline{s} \\ \text{otherwise } i, & \text{where } i = min\{j \in \mathbb{N} \mid s_j = a\} \end{cases}$

Clearly f is a function. For $a \notin \underline{s}$, the answer is unique. Otherwise, the number of indices where a occurs in s is non-zero. The answer in that case is just the minimum index, which is unique.

Resource Specification

- **Data Types**: What data types may be used in the problem and solution specification?
- **Operations**: What operations on the those data types are allowed?
- **Identities**: What identities hold between operations on the data types?

Example: Resource specification to compute factorial

- Data types: natural numbers $\mathbb{N}$
- Operations:
  - decrement $- : \mathbb{N}_+ \to \mathbb{N}$
  - multiplication $* : \mathbb{N}^2 \to \mathbb{N}$
  - comparison $=, \neq : \mathbb{N}^2 \to \mathbb{N}$

A different resource specification to compute factorial

- Data types: integers $\mathbb{Z}$
- Operations:
  - decrement $- : \mathbb{Z} \to \mathbb{Z}$
  - multiplication $* : \mathbb{Z}^2 \to \mathbb{Z}$
  - comparison $=, \neq : \mathbb{Z}^2 \to \mathbb{Z}$

This resource specification is more common in programming languages that have $\mathbb{Z}$ but not $\mathbb{N}$ as primitive datatypes.

Example: Resource specification to compute GCD

- Data types: natural numbers $\mathbb{N}$
- Operations:
  - subtraction $- : \mathbb{N}^2 \to \mathbb{N}$
  - comparison $=, \neq : \mathbb{N}^2 \to \mathbb{N}$

A different resource specification to compute GCD

- Data types: natural numbers $\mathbb{N}$
- Operations:
  - division $/ : \mathbb{N} \times \mathbb{N}_+ \to \mathbb{N}$
  - comparison $=, \neq: \mathbb{N}^2 \to \mathbb{N}$

Standard data types and operations

- Booleans, Natural numbers, Integers, Rationals, enumerated types
- Boolean, Arithmetic and relational operations
- Finite sets and operations on finite sets
- Lists, Stacks, Queues, Finite Trees
- Tupling and projection operations

We will assume that all the above standard data types and operations are available as resources unless specified otherwise.

Non-functional specification

Sometimes there are also **non-functional specification** , e.g.,

- Time available for computation
- Memory space available for computation
- Security, Performance and usability of the solution

Assumption: no non-functional requirements.

Contents

Definition 11 (Mapcode Machine)
A *mapcode machine* is a tuple

$$\mathcal{M} = \langle I, A, X, \rho, F, \pi \rangle$$

consisting of

- an *input set* I
- an *output set* A
- a *state space* X
- an *init map* $\rho : I \to X$,
- a *program map* $F : X \to X$
- an *answer map* $\pi : X \to A$

Definition 12 (Mapcode Algorithm)
Let

$$\mathcal{M} = \langle I, A, X, \rho, F, \pi \rangle$$

be mapcode machine.

$\mathcal{M}$ is an algorithm if

$$\rho(I) \subseteq con(F)$$

Definition 13 (Map computed by a mapcode algorithm)
Let

$$\mathcal{M} = \langle I, A, X, \rho, F, \pi \rangle$$

be a mapcode algorithm.

The map computed by $\mathcal{M}$ is the function

$$\pi \circ F^{\infty} \circ \rho$$

Definition 14 (Computation of a function by a mapcode machine)
Let $f : I_{\mathcal{F}} \to A_{\mathcal{F}}$ be a specification map.

Let $\mathcal{M} = (I, A, X, \rho, F, \pi)$ be a mapcode machine.

$\mathcal{M}$ **computes** f if

- **Signature**: $I_{\mathcal{F}} = I$ and $A_{\mathcal{F}} = A$.
- **Convergence:** $\mathcal{M}$ is an algorithm.
- **Partial Correctness:** Assuming $\mathcal{M}$ is an algorithm, the function computed by $\mathcal{M}$ is f:

$$\pi \circ F^{\infty} \circ \rho = f$$

Mapcode Commute Diagram

$$
\begin{array}{ccc}
I & \xrightarrow{\quad f \quad} & A \\
\Big\downarrow \rho & & \pi \Big\uparrow \\
con(F) & \xrightarrow{F^{\infty}} & fix(F)
\end{array}
$$

Algorithmic Problem Solving for mapcode

Definition 15 (Algorithmic Problem solving via Mapcode)
Let $\mathcal{P} = \langle \mathcal{F}, \mathcal{R}, \mathcal{N} \rangle$ be a a problem with functional specification

$$\mathcal{F} = \langle I_{\mathcal{F}}, A_{\mathcal{F}}, f : I_{\mathcal{F}} \to A_{\mathcal{F}} \rangle$$

A mapcode machine

$$\mathcal{M} = (I, A, X, \rho, F, \pi)$$

is a **solution** to $\mathcal{P}$ if

- **Design:** $\mathcal{M}$ computes f.
- **Implementation:** I, X and A are defined using the datatypes in $\mathcal{R}$. ρ, π and F are defined using the operations in $\mathcal{R}$.
- **Quality:** $\mathcal{M}$ satisfies the non-functional requirements $\mathcal{N}$.

Mapcode machine in Python

```
# Mapcode
# takes
#     rho: I->con(F)
#     F: X->X,
#     pi: fix(F)->A
# returns element in A
def mapcode(rho,F,pi):
    F_infty=limit_map(F)
    def f(v):
        return pi(F_infty(rho(v)))
    return f
```

Contents

Well-founded relations

Let A be a set. Let $<$ be a binary relation on A. $(A, <)$ is **well-founded** if there are no infinite descending chains of the form:

$$\ldots < a_2 < a_1 < a_0$$

Bound Function

Definition 16 (Bound Function)
Let $(W, <)$ be a well-founded relation. Let $D = \langle X, F \rangle$ be a discrete flow. A function $B : X \to W$ is a **bound function** for D if whenever $x \in X$ is transient, $B(F(x)) < B(x)$.

Bound Function implies convergence

Lemma 17 (Bound function implies convergence)

Let $D = (X, F : X \to X)$ be a discrete flow such that there exists a well-founded relation $(W, \leq)$ and a bound function $B : X \to W$ for D. Then X is convergent and $X = con(F)$.

Proof that Bound Function implies convergence

1. Let $B : X \to W$ be a bound function for D.

2. Suppose X is not convergent. Then there is a trajectory $\{a_i\}$ where each $a_i = F^i(a_0)$ is transient.

3. Since B is a bound function and a_i is transient, $B(F(a_i)) = B(a_{i+1}) < B(a_i)$

4. Hence, we have an infinite descending chain

$$\ldots\ B(a_2) < B(a_1) < B(a_0)$$

5. But no such chain is possible since W is well-founded. Contradiction.

Lemma 18 (Convergence for mapcode)
Let $\mathcal{M} = (I, A, X, \rho, F, \pi)$ be a mapcode machine.

Consider the subflow generated by $\rho(I)$:

$D_{orb(\rho(I))} = (orb(\rho(I)), F|_{orb(\rho(I))})$ *of* (X, F) .

If there is a bound function for $D_{orb(\rho(I))}$, then $\rho(I) \subseteq con(F)$

Proof of Convergence in mapcode

Proof:

1. By Lemma 17, $orb(\rho(I))$ is convergent, i.e., $orb(\rho(I)) = con(F|_{\rho(I)})$.

2. But $con(F|_{\rho(I)}) \subseteq con(F)$

3. From steps 1 and 2,

$$\rho(I) \subseteq con(F)$$

Proof Principle for Convergence

$\boxed{\text{To show that } \mathcal{M} = (I, A, \rho, X, F, \pi) \text{ is an algorithm, it suffices to demonstrate a bound function for the flow generated by } \rho(I).}$

Informally, some element of the state has to 'decrease' for each iteration.

Contents

Multiplication using addition

Informal presentation of requirement: Given two natural numbers, compute their product using addition.

- *Functional Specification*
  - $I = \mathbb{N}^2$: The input consists of two natural numbers.
  - $A = \mathbb{N}$: The answer is a natural number
  - $f_* : I \to A$ is the multiplication function: $f(a, b) = a * b$
- *Resource specification*
  - Data types: Natural Numbers
  - Operations: Comparison with zero, decrement and addition

Solution mapcode machine

$$\mathcal{M}_* = \langle \rho : I \to X, F : X \to X, \pi : X \to A \rangle$$

where

- **State Space:** $(x, y, z) \in X = \mathbb{N}^3$
- **Init map:** $\rho(x, y) = (x, y, 0)$
- **Answer map:** $\pi(x, y, z) = z$
- **Dynamical map:**

$$F(x, y, z) = \begin{cases} (x, y, z) & \text{if } x = 0 \\ (x-1, y, z+y) & \text{if } x > 0 \end{cases}$$

Multiplication with mapcode in Python: rho and pi

```
# computes m*n using mapcode
# by repeatedly adding
# n to 0, m times

def rho(i):
    [m,n] = i
    return [m, n, 0]

def pi(x):
    [m, n, a] = x
    return a
```

Multiplication with mapcode in Python: F and f

```
def F(x):
    [m,n,a] = x # m, n, a are naturals
    if m == 0:
        return x
    else:
        return [m-1, n, a+n]


# multiply(m,n) computes m*n
def multiply(m,n): # m,n are naturals
    f = mapcode(rho, F, pi)
    return f([m,n])
```

Computation with $\mathcal{M}_*$

Computing $f(2,3)$:

$$
\begin{tikzcd}
(2,3) \arrow[d, "\rho"] & & 6 \\
(2,3,0) \arrow[r, "F"] & (1,3,3) \arrow[r, "F"] & (0,3,6) \arrow[u, "\pi"']
\end{tikzcd}
$$

Convergence: Showing that $\mathcal{M}_*$ is an algorithm

We wish to prove that

$$\rho(I) \subseteq con(F)$$

Proof: It is easy to verify that $G : X \to \mathbb{N}$ where

$$G(x, y, z) \stackrel{\text{def}}{=} x$$

is a bound function for $\langle X, F \rangle$:

1. Let $w = (x, y, z) \in X$ be transient.
2. Then $x > 0$ (by definition of F)
3. $G(F(x, y, z)) = x - 1 < x = G(x, y, z)$.

Correctness of $\mathcal{M}_*$ (Informally)

We have

$$(x, y, 0) \xrightarrow{F} (x-1, y, y) \xrightarrow{F} (x-2, y, 2y) \xrightarrow{F} \ldots \xrightarrow{F} (x-x, y, xy)$$
$$= (0, y, xy)$$

Hence, we may conjecture that

$$F^\infty(x, y, 0) = (0, y, xy)$$

Then

$$\begin{aligned} \pi(F^\infty(\rho(x, y))) &= \pi(F^\infty(x, y, 0)) \\ &= \pi(0, y, xy) \\ &= xy \end{aligned}$$

Contents

Invariant Function

Definition 19 (Invariant Function)

- Let $D = (X, F : X \to X)$ be a discrete flow.
- Let E be any set.
- A function $\theta : X \to E$ is an **invariant function** for D if, for each $x \in X$,

$$\theta(x) = \theta(F(x))$$

Invariant Function, iterates and limit map

Lemma 20 (Invariant function and iterates)
Let $D = (X, F)$ be a discrete flow. Let $\theta : X \to E$ be an invariant function for D.

Then, for each $x \in X$ and for each $i \in \mathbb{N}$,

$$\theta(x) = \theta(F^n(x))$$

Corollary 21

If $x \in con(F)$, then

$$\theta(x) = \theta(F^\infty(x))$$

Theorem 22 (Partial Correctness via invariant function)

- *Consider a specification map* $f : I \to A$ *and a mapcode algorithm* $\mathcal{M} = \langle I, X, A, \rho : I \to X, F : X \to X, \pi : X \to A \rangle$
- *Let* $\theta : X \to A$ *be an invariant map for* (X, F)*. Assume*
  - **init:** $\theta(\rho(i)) = f(i)$ *for each problem instance* $i \in I$ *and*
  - **answer:** $\theta(x) = \pi(x)$ *for each fixed point* $x \in \mathit{fix}(F)$*.*
- *Then,* $\pi \circ F^\infty \circ \rho = f$*.*

Proof of partial correctness via invariant function (short version)

$$\begin{aligned} f(i) &= \theta(\rho(i)) & (1) \\ &= \theta(F^{\infty}(\rho(i))) & (2) \\ &= \pi(F^{\infty}(\rho(i))) & (3) \end{aligned}$$

Proof of partial correctness via invariant function

$$
\begin{aligned}
\rho(I) &\subseteq con(F) && \mathcal{M}\text{ algorithm: Given} && (4)\\
s &\in I && \text{assumption} && (5)\\
\theta(\rho(s)) &= f(s) && \text{from `init': Given} && (6)\\
\rho(s) &\in con(F) && \text{from 4 and 5} && (7)\\
F^{\infty}(\rho(s)) &\in \mathit{fix}(F) && \text{from 7 and defn of } F^{\infty} && (8)\\
\pi(z) &= \theta(z) && \forall z \in \mathit{fix}(F)\text{, `answer': Given} && (9)\\
\pi(F^{\infty}(\rho(s))) &= \theta(F^{\infty}(\rho(s))) && \text{from 8 and 9} && (10)\\
&= \theta(\rho(s)) && \text{since } \theta \text{ is invariant} && (11)\\
&= f(s) && \text{from 6} && (12)
\end{aligned}
$$

Proof Principle for partial correctness

To prove that a mapcode algorithm $\mathcal{M} = (I, A, X, \rho, F, \pi)$ computes a specification map, $f : I \to A$, it is sufficient to construct an invariant function $\theta : X \to E$ such that the init and answer conditions are met:

1. $\theta(\rho(i)) = f(i)$ for each $i \in I$
2. $\theta(x) = \pi(x)$ for each $x \in fix(F)$

Contents

Invariant function for $\mathcal{M}_*$

For $\mathcal{M}_*$, $I = \mathbb{N}^2$, $A = \mathbb{N}$ and $X = \mathbb{N}^3$, $f_*(x,y) = xy$, $\rho(x,y) = (x,y,z)$ and $\pi(x,y,z) = z$.

Define

$$\boxed{\theta(x,y,z) \stackrel{\text{def}}{=} xy + z}$$

Intuition:

- z: work already done
- xy: work yet to be done (x number of y's need to be added)
- $\theta(x,y,z)$: total work derived by combining work done and work that needs to be done.

Verify that θ satisfies init condition

init: If $(x, y) \in I$, then $\theta(\rho(x, y)) = f_*(x, y)$

- $\theta(\rho(x, y)) = \theta(x, y, 0) = xy + 0 = xy$
- $f_*(x, y) = xy$

Verify that θ satisfies answer condition

answer: If $(x, y, z) \in \mathit{fix}(F)$, then $\pi(x, y, z) = \theta(x, y, z)$

- Since $(x, y, z) \in \mathit{fix}(F)$, $x = 0$
- $\pi(x, y, z) = z$
- $\theta(x, y, z) = \theta(0, y, z) = 0 * y + z = z$

Total Correctness: $\mathcal{M}_*$ computes f_*

- **Convergence:** $\rho(I) \subseteq con(F)$: verified by constructing suitable bound function in Slide 70.

- **Partial Correctness:** $\rho(I) \subseteq con(F) \implies \pi \circ F^\infty \rho = f_*$: verified by constructing suitable invariant function in Slide 80.

This proves

Total Correctness: $\mathcal{M}_*$ computes f_*.

Example: Factorial

- The problem is $\mathcal{P} = \langle I = \mathbb{N}, A = \mathbb{N}, f(n) = !n, P = \{-, *\}\rangle$

- Mapcode machine:
  $\mathcal{M}_! = \langle \rho : I \to X, F : X \to X, \pi : X \to A\rangle$ where
  1. $X = \mathbb{N}^2$
  2. $\rho(n) = (n, 1)$
  3. $F(i, a) = \begin{cases} (i, a) & \text{if } i = 0 \\ (i-1, a*i) & \text{otherwise} \end{cases}$
  4. $\pi(i, a) = a$

Factorial: bound function

$$G : X \to \mathbb{N}, G(i,a) \stackrel{\text{def}}{=} i$$

is a bound function for $\langle X, F \rangle$. Verify:

- Let $(i,a) \in X$ be transient.
- Then, $i > 0$, and
- $G(F(i,a)) = i - 1 < i = G(i,a)$.

Factorial: invariant function

$$\theta(i, a) \stackrel{\text{def}}{=} a * i!$$

- Let $x = (i, a) \in X$
- case: $x \in \mathit{fix}(F)$, then clearly $\theta(x) = \theta(F(x))$.
- If $x \notin \mathit{fix}(F)$, then $i > 0$:
- $\theta(i, a) = a * i! = a * i * (i - 1)! \ = \theta(F(i, a))$.

init and answer conditions for θ

- Init: $\theta(\rho(n)) = \theta(n, 1) = 1 * n! = n!$
- Answer: Let $(i, a) \in \mathit{fix}(F)$. Then $i = 0$
  - $\pi(0, a) = a$ and
  - $\theta(0, a) = a * 0! = a$

Partial Correctness: $\rho(I) \subseteq A \implies \pi \circ F^{\infty} \circ \rho = !$

- **Convergence:** $\rho(I) \subseteq con(F)$: Shown in Slide 85
- **Partial Correctness**: $\rho(I) \subseteq A \implies \pi \circ F^{\infty} \circ \rho = !$: Shown in Slide 87

Hence $\mathcal{M}_!$ computes !.

Exercise: Invariant functions and mapcode machines

- Summation: Define an invariant function.
- GCD: Define a mapcode machine to compute gcd.
- max: Define a mapcode machine to compute the maximum element of a nonempty list of natural numbers.

Contents

Concepts

- Problem
- Functional Specification
- Mapcode Machine
- Convergence, Partial and Total correctness
- Computation of a function by a mapcode machine
- Invariant function
- Init and answer conditions

References