Mapcode: a dynamical systems approach to algorithmic program development

Venkatesh Choppella
IIIT Hyderabad

23rd December 2022
ACM India CS Ed Workshop, IIT Gandhinagar India

Abstract

Mapcode is a methodology for iterative algorithmic problem solving introduced by Kasturi Viswanath in his book An Introduction to Mathematical Computer Science (Universities Press, 2008). In Mapcode, the requirement and design of an algorithm are expressed as a collection of maps (total functions). Once the design is available, it can be easily coded as a program using the mapcode combinator.

The workshop is an introduction to program development using the mapcode methodology. This will be done through a series of examples which illustrate the mapcode approach to the specification, design and coding of solutions as programs.

Contents

1 Mapcode

Mapcode: A Dynamical Systems Approach to Algorithmic Program Development

ACM CS Ed Workshop, IIT Gandhinagar, 23rd December 2022

Venkatesh Choppella

IIIT Hyderabad

Contents

What is Mapcode?

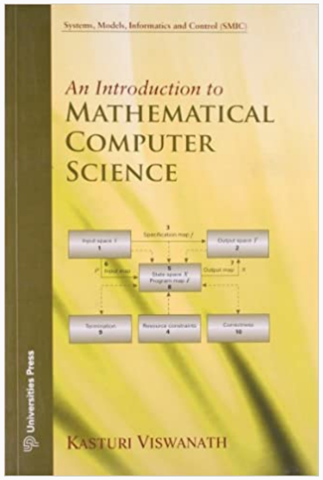
1. A *practical* theory of computing (vocabulary, concepts, notation)
2. aimed at modular design of sequential algorithms using **maps**,
3. and their implementation as **code**.

Mapcode is for Programmers

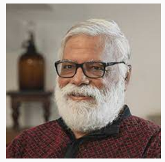
Mapcode allows programmers

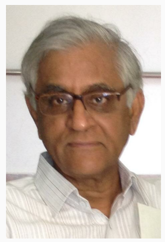
1. to talk and reason about their programs
2. to structure their programs in a modular way
3. to build reusable libraries

The Mapcode book



An Introduction to Mathematical Computer Science. Kasturi Viswanath. Universities Press 2008. (Foreword by Kesav Nori)


KV Nori


K Viswanath

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

The scientific method

- **Identifying** *observables*
- **Tracing** changes over time
- **Collecting** different traces into *behaviour*
- **Explaining** behaviour by **generating** it from a dynamical *model*
- **Predicting** new behaviour by running models
- **Refining** the model to account for discrepancies between real and predicted behaviour

Traces over an Observation Space

Observation Space Y: set of *observables*

1. $Y = \mathbb{N}$ $\quad evens = [0 \to 2 \to 4 \to 6 \ldots]$
2. $Y = \mathbb{R}$ $\quad asset = [100.0 \to 110.0 \to 121.0 \to 132.1 \ldots]$
3. $Y = \mathbb{Q}$ $\quad zeno = [1 \to 1/2 \to 1/4 \to 1/8 \to \ldots]$
4. $Y = \{red, green, yellow\}$
   $trafficLight = [\text{red} \to \text{green} \to \text{yellow} \to \text{red} \ldots]$

Traces map naturals to observations

$$trace : \mathbb{N} \to Y$$

1. *evens*: $$evens_i = 2i$$
2. *asset*: $$asset_i = 100 \times (1.1)^i$$
3. *zeno*: $$zeno_i = 2^{-i}$$
4. *trafficLight* :

$$trafficLight_i = \begin{cases} \text{red} & \text{if } i(mod\ 3) = 0 \\ \text{green} & \text{if } i(mod\ 3) = 1 \\ \text{yellow} & \text{if } i(mod\ 3) = 2 \end{cases}$$

Generating Traces

Can we *generate* a trace?

1. Identify a **state space** X
2. Start from an **initial state** x_0
3. Apply a **dynamical map** F to transform the current state x to the next state x'. $\boxed{x' = F(x)}$
4. Repeat indefinitely to yield a **trajectory**

$$trj(x_0) = [x_0 \to F(x_0) \to F^2(x_0) \to \ldots]$$

5. Project the trace from the trajectory

State and Dynamical Map

1. evens: $x_0 = 0,\ F(x) = x + 2$
2. asset: $x_0 = 100,\ F(x) = 1.1 \times x$
3. zeno: $x_0 = 1,\ F(x) = x/2$
4. trafficLight: $x_0 = \text{red}$,

$$F(x) = \begin{cases} \text{green} & \text{if } x = \text{red} \\ \text{yellow} & \text{if } x = \text{green} \\ \text{red} & \text{if } x = \text{yellow} \end{cases}$$

Traces, take 2

1. *squares* $= [0 \to 1 \to 4 \to 9 \ldots]$
2. *fac4* $= [1 \to 4 \to 12 \to 24 \to 24 \to 24 \ldots]$
3. *pingala* $= [0 \to 1 \to 1 \to 2 \to 3 \to 5 \ldots]$
4. *bubblesort* $= [8\ 6\ 9\ 7] \to [6\ 8\ 9\ 7] \to [6\ 8\ 9\ 7] \to [6\ 8\ 7\ 9] \to [6\ 8\ 7\ 9] \to [6\ 7\ 8\ 9] \to [6\ 7\ 8\ 9] \ldots$

Generating squares: State vs Observation

$$\begin{tikzcd}
\textit{Trajectory} & (1,0) \arrow[r] \arrow[d, dashed, "h"] & (3,1) \arrow[r] \arrow[d, dashed, "h"] & (5,4) \arrow[r] \arrow[d, dashed, "h"] & (7,9) \arrow[d, dashed, "h"] \\
\textit{Trace} & 0 \arrow[r] & 1 \arrow[r] & 4 \arrow[r] & 9
\end{tikzcd}$$

Display Map projects an observation from the state

- State Space X
- Observation space Y

Display map

$$h : X \to Y$$

Squares: Display and Dynamical Map

1. Observation Space $Y = \mathbb{N}$
2. State Space $X = \mathbb{N} \times \mathbb{N}$

   state vector $x = (v, d)$

   initial state $x_0 = (1, 0)$
3. Display Map $h : X \to X$

$$h(v, d) = d$$

4. Dynamical Map $F : X \to X$

$$F(v, d) = (v + 2, d + v)$$

$$\begin{tikzcd}
(1,0) \arrow[r, dashed, "h"] \arrow[d] & 0 \arrow[d] \\
(3,1) \arrow[r, dashed, "h"] \arrow[d] & 1 \arrow[d] \\
(5,4) \arrow[r, dashed, "h"] \arrow[d] & 4 \arrow[d] \\
(7,9) \arrow[r, dashed, "h"] \arrow[d] & 9 \arrow[d] \\
\vdots & \vdots
\end{tikzcd}$$

Fac4: Display and Dynamical Map

1. Observation Space $Y = \mathbb{N}$
2. State Space $X = \mathbb{N} \times \mathbb{N}$

   state vector $x = (i, a)$

   initial state $x_0 = (4, 1)$
3. Display map $h : X \to X$

$$h(i, a) = a$$

4. Dynamical Map $F : X \to X$

$$F(i, a) = \begin{cases} (i, a) & \text{if } i = 0 \\ (i - 1, a * i) & \text{otherwise} \end{cases}$$

$$\begin{tikzcd}
(4,1) \arrow[r, dashed, "h"] \arrow[d] & 1 \arrow[d] \\
(3,4) \arrow[r, dashed, "h"] \arrow[d] & 4 \arrow[d] \\
(2,12) \arrow[r, dashed, "h"] \arrow[d] & 12 \arrow[d] \\
(1,24) \arrow[r, dashed, "h"] \arrow[d] & 24 \arrow[d] \\
(0,24) \arrow[r, dashed, "h"] \arrow[d] & 24 \arrow[d] \\
(0,24) \arrow[r, dashed, "h"] \arrow[d] & 24 \arrow[d] \\
{} & {}
\end{tikzcd}$$

Summary of Spaces and Maps

- Observation Space: Y
- State Space: X
- Trace: $\mathbb{N} \to Y$
- Trajectory: $X \to (\mathbb{N} \to X)$
- Display Map: $h : X \to Y$
- Dynamical Map: $F : X \to X$
- Iterative System: (X, F) (also called **discrete flow**)

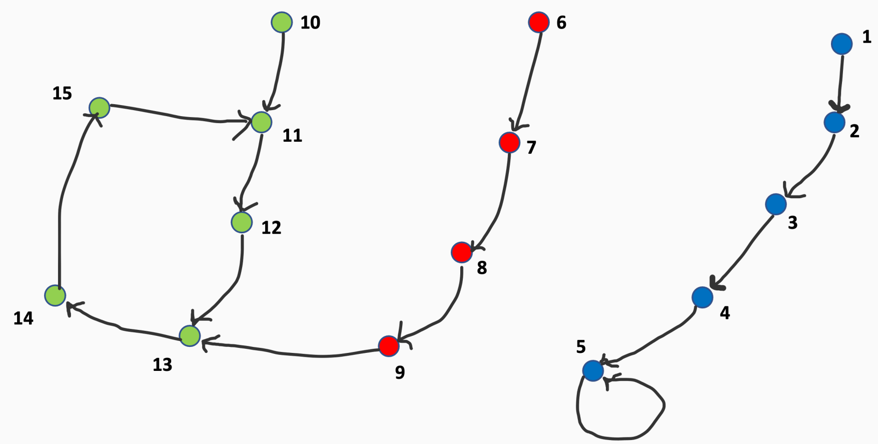
Exercise: Construct examples of Iterative Systems

Give two examples of

1. An observation space and a trace over it.
2. An iterative system and an initial state
3. A display map that connects the trajectory of the initial state to the trace.

Contents

Iterative Systems in pictures



Trajectories and Orbits

Let $(X, F : X \to X)$ be an iterative system

Trajectory of x

$$trj(x) = [x, \quad F(x), \quad F(F(x)), \quad F^3(x), \quad \ldots]$$

Orbit of x

$$orb(x) = \{x, \quad F(x), \quad F(F(x)), \quad F^3(x), \quad \ldots\}$$

Point types

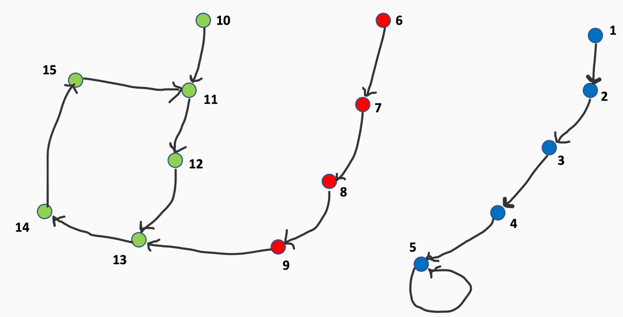
1. **Fixed** : $x = F(x)$
2. **Transient**: $x \neq F(x)$
3. **(Finitely) Convergent**: x reaches a fixed point after a finite number of applications of F.

$fix(F)$ and $con(F)$

$fix(F)$: set of all fixed points of F

$con(F)$: set of all convergent points of F

Example of fixed, transient and convergent points



Fixed: 5

Transient: all points except 5

Convergent: all blue points

Exercise: Identify fixed, transient and convergent points

Identify the fixed, transient and convergent points of the dynamical maps in the

1. Squares example
2. Fac4 example

Contents

Iterating to a fixed point

1. How to transform the state at each step:

$$x := F(x)$$

2. When to stop: **fixed point**

$$x = F(x)$$

Fixed Point iteration

```
# loop: [X, X->X] -> X
# loop(x,F) assumes
# x is in con(F)
# returns the fixed point
# reached by x

def loop(x,F):
    while (x != F(x)):   # x is transient
        x = F(x)         # update x
    return x
# x is a fixed point
```


Limit Map takes convergent points to fixed points

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

$$F^\infty(x) = \lim_{n\to\infty} F^n(x)$$

for $x \in con(F)$

Limit Map implementation

```
from fpi import loop

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

Example: Limit Map for Factorial

1. $fix(F) = \{0\} \times \mathbb{N}$
2. $con(F) = \mathbb{N} \times \mathbb{N}$
3. $F^\infty : con(F) \to fix(F)$

$$F^\infty(i, a) = a * i!$$

Exercise: Prove this using induction on i.

Contents

Components of Algorithmic Problem Solving

1. specify problem: what to compute?
2. assume datatypes and primitive operations: what to compute with?
3. design solution: how to compute?

Instance Space

Mapcode Step 1: What is the type of the problem instance?

Instance Space I

Answer Space

Mapcode Step 2: What is the type of the answer?

Answer Space A

Specification Map

: **Mapcode Step 3**: What is (the map) to be computed?

Specification Map $f : I \to A$

Machine Datatypes and Primitive Operations

Mapcode Step 4: What are the primitive data types and operations available when designing the solution?

Example: Problem Specification for Factorial

Compute Factorial using subtraction and multiplication

1. Instance Space $I = \mathbb{N}$
2. Answer Space $A = \mathbb{N}$
3. Specification Map $f(n) = n!$
4. Machine datatypes: natural numbers $\mathbb{N}$
5. Primitive operations: decrement, mulitiply

Exercise: Write Problem Specifications

Write down the problem specifications for the following informal requirements

1. Multiplication: Compute the product of two natural numbers using subtraction and addition.

2. GCD: Compute the greatest common divisor of two natural numbers using subtraction and comparison

Contents

State Space

Mapcode Step 5: What are the program variables and their types?

State space X

Example: Statespace for Factorial

State Space $X = \mathbb{N} \times \mathbb{N}$

state vector $x = (i, a)$ (index, accumulator)

Exercise: Specify Statespaces

Specify statespaces for

1. Multiplication problem

2. GCD problem

Init map

Mapcode Step 6: How does one take an instance and map it to an initial state?

Init map $\rho : I \to X$

Answer Map

Mapcode Step 7: How does one extract the answer from a (final) state?

Answer map $\pi : X \to A$

Init and Answer maps for Factorial

Instance space $I = \mathbb{N}$

Answer space $A = \mathbb{N}$

State space $X = \mathbb{N} \times \mathbb{N}$

Init map $\rho : I \to X$

$$\rho(n) = (n, 1)$$

Answer map $\pi : X \to A$

$$\pi(i, a) = a$$

Program Map

Mapcode Step 8: What is the rule to update the program variables at each step?

$$\textbf{Program map} \quad F : X \to X$$

- aka **dynamical map**.
- F **acts** on X.

Example: Program Map for Factorial

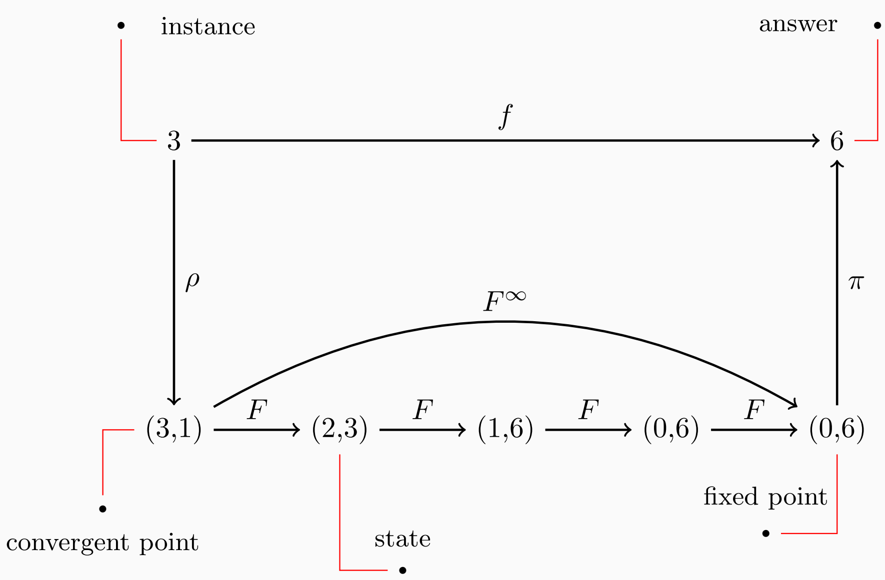
$X = \mathbb{N} \times \mathbb{N}$

$x = (i, a)$

$F : X \to X$

$$F(i, a) = \begin{cases} (i, a) & \text{if } i = 0 \\ (i - 1, a * i) & \text{otherwise} \end{cases}$$

Execution Diagram for computing 3!



Exercise: Program maps and execution diagrams

Specify the program maps for

1. Multiplication problem
2. GCD problem

Also draw execution diagrams for computing

1. 2×3
2. $gcd(12, 8)$

Mapcode Machine

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

$\mathcal{M}$ is an *algorithm* if $\rho(I) \subseteq con(F)$, i.e., if for each instance i, $\rho(i)$ is a convergent state.

If $\mathcal{M}$ is an algorithm then $\rho; F^{\infty}; \pi$ is the map *computed by* $\mathcal{M}$.

Problem solving via Mapcode

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

$\mathcal{M}$ is an *algorithm* if $\rho(I) \subseteq con(F)$, i.e., if for each instance i, $\rho(i)$ is a convergent state.

If $\mathcal{M}$ is an algorithm then $\rho; F^{\infty}; \pi$ is the map *computed by* $\mathcal{M}$.

Mapcode Checklist so far

No.	Mapcode artefact	Notation
1.	Instance space	I
2.	Answer space	A
3.	Specification map	$f : I \to A$
4.	Primitive Operations	
5.	State space	X
6.	Init map	$\rho : I \to X$
7.	Answer map	$\pi : X \to A$
8.	Program map	$F : X \to X$

Contents

The mapcode program generator

```
from fpi import limit_map

# F: X->X, rho: I->X, pi: X->A
# rho(I) subset of con(F)

# mapcode(rho, F, pi): I -> A
def mapcode(rho,F,pi):
    F_inf=limit_map(F)
    def f(i):
        return pi(F_inf(rho(i)))
    return f
```


Mapcode machine for Factorial

```
def rho(n):
    return ([n,1])

def pi(x):
    [i,a] = x
    return a

def F(x):
    [i, a] = x
    if (i == 0):
        return x
    else:
        return [i-1, a*i]
```


Computing factorial using the mapcode machine

```
from mapcode import mapcode

fact = mapcode(rho, F, pi)
```


Exercise: Specifying mapcode machines

Implement ρ, π and F and then use the mapcode library function to implement a solution for the following problems

1. Multiplication problem

2. GCD problem

Contents

Finite Convergence (Termination)

$$\begin{tikzcd} I \arrow[r, "f"] \arrow[d, "\rho"] & A \\ con(F) \arrow[r, "F^\infty"] & fix(F) \arrow[u, "\pi"'] \end{tikzcd}$$

Mapcode Step 9: How do we know that ρ maps each instance to a convergent point?

prove

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

Partial Correctness

$$\begin{tikzcd} I \arrow[r, "f"] \arrow[d, "\rho"] & A \\ con(F) \arrow[r, "F^\infty"] & fix(F) \arrow[u, "\pi"'] \end{tikzcd}$$

Mapcode Step 10: Assuming termination, how do we know that the diagram commutes?

prove

$$f = \rho; F^\infty; \pi$$

Contents

Motivation

- We want to exploit discrete flows and iterative systems to compute answers.
- The way we wish to do this to construct convergent trajectories.
- The answer is 'embedded' in a fixed point.
- However, not all states may converge to fixed points.
- Therefore, we are obligated to demonstrate that the initial state we choose indeed converges.
- In such a case, the computation is said to converge.

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:

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

Examples and non-examples of well-founded relation

- $\langle \mathbb{N}, < \rangle$ (natural numbers and 'less than') is well-founded.
- $\langle \mathbb{Z}, < \rangle$ (integers and 'less than') is not well-founded.
- $\langle \mathbb{Q}, < \rangle$ (rational numbers and 'less than') is not well-founded.

Bound Map

Definition 1 (Bound Map)

Let $(W, <)$ be a well-founded relation. Let $D = \langle X, F \rangle$ be a discrete flow.

A map $B : X \to W$ is a **bound map** for D if whenever $x \in X$ is transient, $B(F(x)) < B(x)$.

Bound Maps: Example 1

- Let $D = \langle \mathbb{N}, F : \mathbb{N} \to \mathbb{N} \rangle$ be a flow, where
$F(x) = \begin{cases} 0 & \text{if } x = 0 \\ x - 1 & \text{otherwise} \end{cases}$.
- Then the identity function is a bound map for D.

Bound Maps: Example 2

- Let $D = \langle \mathbb{N}, F : \mathbb{N} \to \mathbb{N} \rangle$ where
$F(x) = \begin{cases} k & \text{if } x \geq k \\ x + 1 & \text{otherwise} \end{cases}$.
- Let $B : X \to \mathbb{N}$ where $B(x) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } x = k \\ 1 & \text{if } x > k \\ k - x & \text{otherwise} \end{cases}$
- B is a bound map for D. (Verify this.)

Bound Map implies convergence

Lemma 2 (Bound map implies convergence)

Let $D = (X, F : X \to X)$ be a discrete flow such that there is a bound map for D. Then every element of X is convergent.

Proof that Bound Map implies convergence

1. Let $B : X \to W$ be a bound map 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. For each i, $B(F(a_i)) = B(a_{i+1}) < B(a_i) = B(a_i)$
4. Hence, we have an infinite descending chain
$$\ldots < B(a_1) < B(a_0)$$
5. But no such chain is possible since $(W, <)$ is well-founded. Contradiction.

Convergence in mapcode

Lemma 3 (Convergence for mapcode)

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

Let $S = \rho(I)$. Consider the subflow $D_{orb(S)} = (orb(S), F|_{orb(S)})$ of (X, F).

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

Proof of convergence in mapcode

Proof:

1. By Lemma 2, $orb(S)$ is convergent, i.e., $orb(S) \subseteq con(F)$.
2. From step 1 and the fact that $S \subseteq orb(S)$, it follows that $S = \rho(I) \subseteq con(F)$.

Proof Principle for Convergence

To show that $\mathcal{M}$ is an algorithm, it suffices to demonstrate a bound map for the flow generated by $\rho(I)$.

Contents

Invariant Map

Definition 4 (Invariant Map)

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

Invariant map, iterates and limit map

Lemma 5 (Invariant map and iterates)

Let $D = (X, F)$ be a discrete flow. Let $\theta : X \to E$ be an invariant map for D.

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

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

Corollary 6

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

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

Partial Correctness via invariant map

Theorem 7 (Partial Correctness via invariant map)

- *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 instance $i \in I$ and*
  - **answer:** $\theta(x) = \pi(x)$ *for each fixed point $x \in fix(F)$.*
- *Then, $\rho; F^\infty \circ \pi = f$.*

Proof of partial correctness via invariant map

$$\begin{aligned}
\rho(I) &\subseteq con(F) && \mathcal{M} \text{ algorithm: Given} && (1)\\
s &\in I && \text{assumption} && (2)\\
\theta(\rho(s)) &= f(s) && \text{from `init': Given} && (3)\\
\rho(s) &\in con(F) && \text{from 1 and 2} && (4)\\
F^\infty(\rho(s)) &\in fix(F) && \text{from 4 and defn of } F^\infty && (5)\\
\pi(z) &= \theta(z) && \forall z \in fix(F), \text{ `answer': Given} && (6)\\
\pi(F^\infty(\rho(s))) &= \theta(F^\infty(\rho(s))) && \text{from 5 and 6} && (7)\\
&= \theta(\rho(s)) && \text{since } \theta \text{ is invariant} && (8)\\
&= f(s) && \text{from 3} && (9)
\end{aligned}$$

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 map $\theta : X \to A$ such that the init and answer conditions are met:

1. **init:** $\theta(\rho(i)) = f(i)$ for each $i \in I$

2. **answer:** $\theta(x) = \pi(x)$ for each $x \in fix(F)$

Bound Map for Factorial

- $I = \mathbb{N}$, $A = \mathbb{N}$, $X = \mathbb{N}^2$, $\rho(n) = (n, 1)$

- $\rho(I) = \mathbb{N} \times \{1\}$

- $F(i, a) = (i, a)$ if $i = 0$, $(i - 1, a * i)$, otherwise

- Let $(W = \mathbb{N}, <)$ denote the usual 'less than' ordering on

Let $B : X \to \mathbb{N}$ be defined as $B(i, a) = i$. Then $B|_{orb(\rho(I))}$ is a bound map for the flow $(orb(\rho(I)), F|_{orb(\rho(I))})$. (Exercise, verify this.)

Invariant Map for Factorial

- $I = \mathbb{N}$, $A = \mathbb{N}$, $X = \mathbb{N}^2$, $\rho(n) = (n, 1)$

- $\rho(I) = \mathbb{N} \times \{1\}$

- $F(i, a) = (i, a)$ if $i = 0$, $(i - 1, a * i)$, otherwise

Let $\theta : X \to A$ be defined as $\theta(i, a) = i! * a$. Then θ is an invariant map, since

- Assume $n \in I$. Then $\theta(\rho(n)) = \theta(n, 1) = n! = f(n)$

- $(i, a) \in fix(F)$. Then $i = 0$,

$$\begin{aligned} \theta(0, a) &= 0! * a \\ &= a \\ &= \pi(0, a) \end{aligned}$$

Exercise: Bound and Invariant maps for Multiplication and GCD

Define bound and invariant maps for

1. multiplication and

2. GCD

Final mapcode checklist

No.	Mapcode artefact	Notation
1.	Instance space	I
2.	Answer space	A
3.	Specification map	$f : I \to A$
4.	Primitive Operations	
5.	State space	X
6.	Init map	$\rho : I \to X$
7.	Answer map	$\pi : X \to A$
8.	Program map	$F : X \to X$
9.	Termination condition	$\rho(I) \subseteq con(F)$
10.	Partial Correctness condition	$f = \rho; F^{\infty}; \pi$

Contents

Homework Exercises

Write literate programs based on mapcode for the following:

1. Finding the maximum of a nonempty array of numbers.
2. Factorial using a stack ('recursion')
3. Addition using increment
4. Bubblesort
5. Finding the height of a binary tree.

Acknowledgments

Thank You All

1. **Neeldhara Misra**: Workshop organization and invitation
2. **Mrityunjay Kumar**: Examples, code
3. **K Viswanath**: General discussions about Mapcode
4. **ACM India**: Travel support
5. **Audience**: Participation!

venkatesh.choppella@iiit.ac.in

Submission and Feedback

https://bit.ly/mapcode2022