

Art (?) and Fun (!) with Formal Methods

Nikolay Shilov (Innopolis University)

talk at PλC,

Rostov-on-Don, April 4, 2017

Part I

WHY I COUNT ON POPULAR SCIENCE

What is wrong with Formal Methods

- Recently David L. Parnas have called (in the paper “Really Rethinking *Formal Methods*”) to question the well-known current formal software development methods why they have not been widely adopted in industry and what should be changed.

In my (not-)humble opinion...

- Industrial applications of Formal Methods are not the unique measure of success.
- Another dimension where we can discuss utility of Formal Methods could be better education.

In my (not-)humble opinion...

- A very popular (in Russia) aphorism of Mikhail Lomonosov (the first Russian academician) says: *Mathematics should be learned just because it disciplines and bring up the mind.*
- I do believe that Formal Methods discipline and bring up minds in Computer Science.

In my (not-)humble opinion...

- A part of the reason of student's and engineer's poor attitude to Formal Methods, is very simple: FM-experts do not care about primary education in the field at the early stage of higher education.

In my (not-)humble opinion...

- In particular, many courses on Formal Semantics start with fearful terms like *state machine*, *logic inference*, *denotational semantics*, etc., without elementary explanations of the basic notions.

Why this talk?

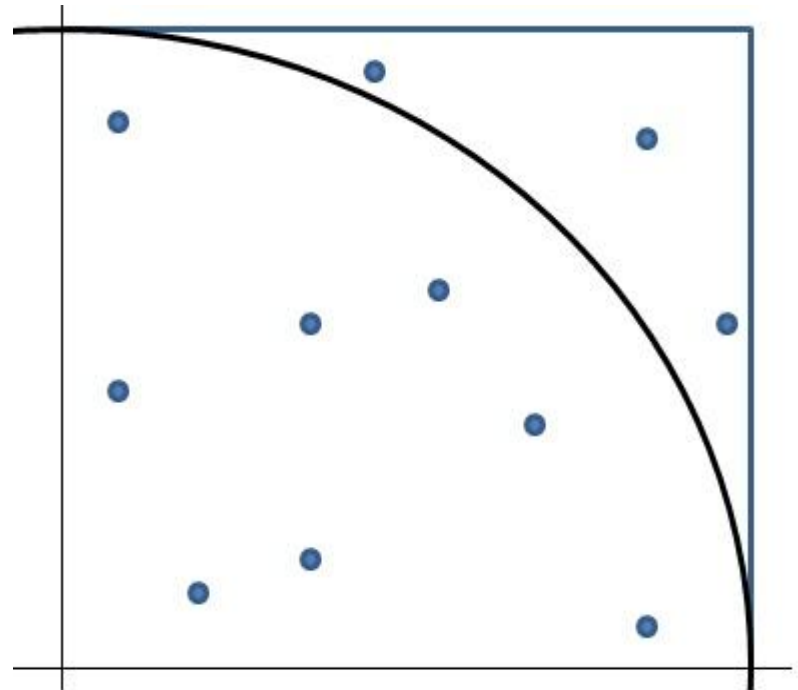
- I would like to present some examples that (I believe) may help to attract attention of undergraduate students to study of Formal Methods.

Part II

WHY MANUAL PROOF AND NUMERIC SIMULATION ARE NOT ENOUGH

MonteCarlo.c

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
int main(void){
    srand(time(NULL));
    int i, j, r, n = 10;
    float pi_val, x, y;
    int n_hits, n_trials=1000000;
    for(j = 0; j < n; j++){n_hits=0;
        for(i = 0; i<n_trials; i++){
            r = rand()% 100000000;
            x = r/100000000.0;
            r = rand()% 100000000;
            y = r/100000000.0;
            if(x*x + y*y < 1.0) n_hits++;}
        pi_val = 4.0*n_hits/(float)n_trials;
    printf("%f \n", pi_val); } return 0;}
```



Experiment

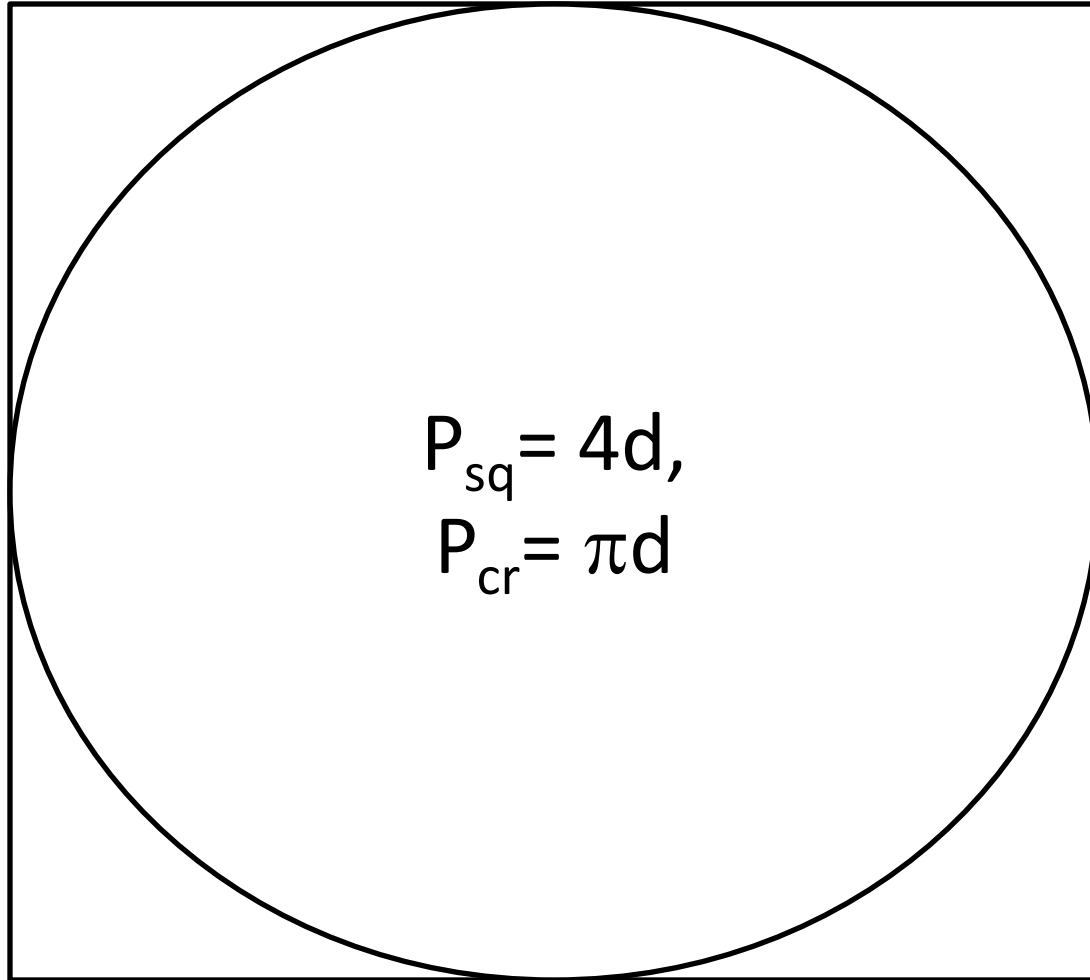
The screenshot displays the Code::Blocks IDE interface. The main editor window shows the source code for `main.cpp`, which implements a Monte Carlo method to estimate the value of pi. The code includes headers for `stdio.h`, `time.h`, and `stdlib.h`. It defines a `main` function that sets a random seed, initializes variables for iterations (`n = 10`), pi value, and coordinates (`x, y`). A nested loop structure performs 10 trials, each with 1,000,000 random points. For each point, it checks if $x^2 + y^2 < 1.0$ and increments the hit counter. The final pi value is calculated as $4.0 * n_hits / n_trials$.

```
1 #include <stdio.h>
2 #include <time.h>
3 #include <stdlib.h>
4 int main(void){
5     srand(time(NULL));
6     int i, j, r, n = 10;
7     float pi_val, x, y;
8     int n_hits, n_trials=1000000;
9     for(j = 0; j < n; j++){n_hits=0;
10         for(i = 0; i<n_trials; i++){
11             r = rand()% 10000000;
12             x = r/10000000.0;
13             r = rand()% 10000000;
14             y = r/10000000.0;
15             if(x*x + y*y < 1.0) n_hits++;}
16         pi_val = 4.0*n_hits/(float)n_trials;
17     printf("%g \n", pi_val); } return 0;}
18
```

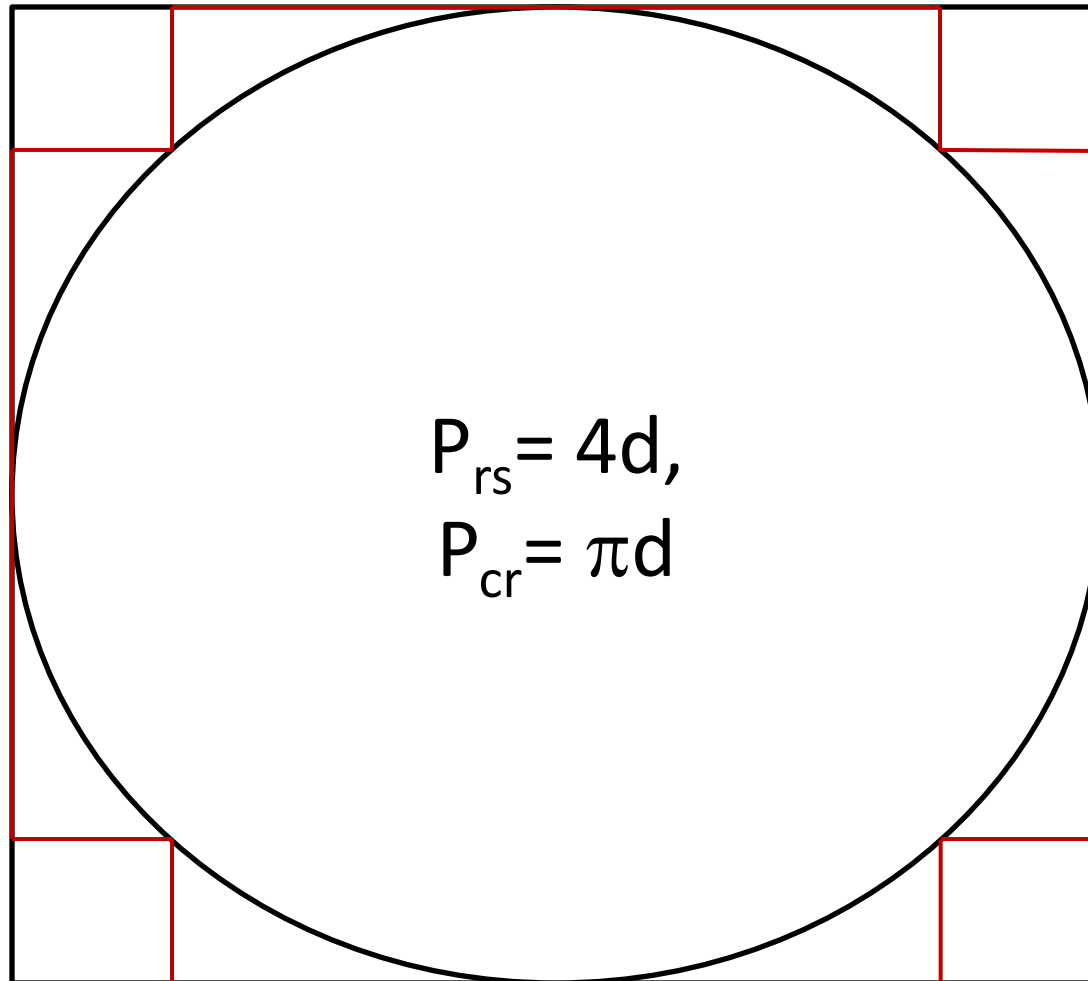
An overlaid console window titled `C:\TryC\PiTrials\bin\Debug\PiTrials.exe` shows the output of the program, which consists of ten lines of the value `4.000000`. Below the output, it displays `Process returned 0 (0x0) execution time : 0.342 s` and `Press any key to continue.`

The bottom status bar of the IDE shows the current window title as `main.cpp [PiTrials] - Code::Blocks 12.11` and the status `WINDOWS-1252 Line 18, Column 1 Insert Read/Write default`. The system tray at the bottom right indicates the date and time as `4:16 PM 3/13/2014`.

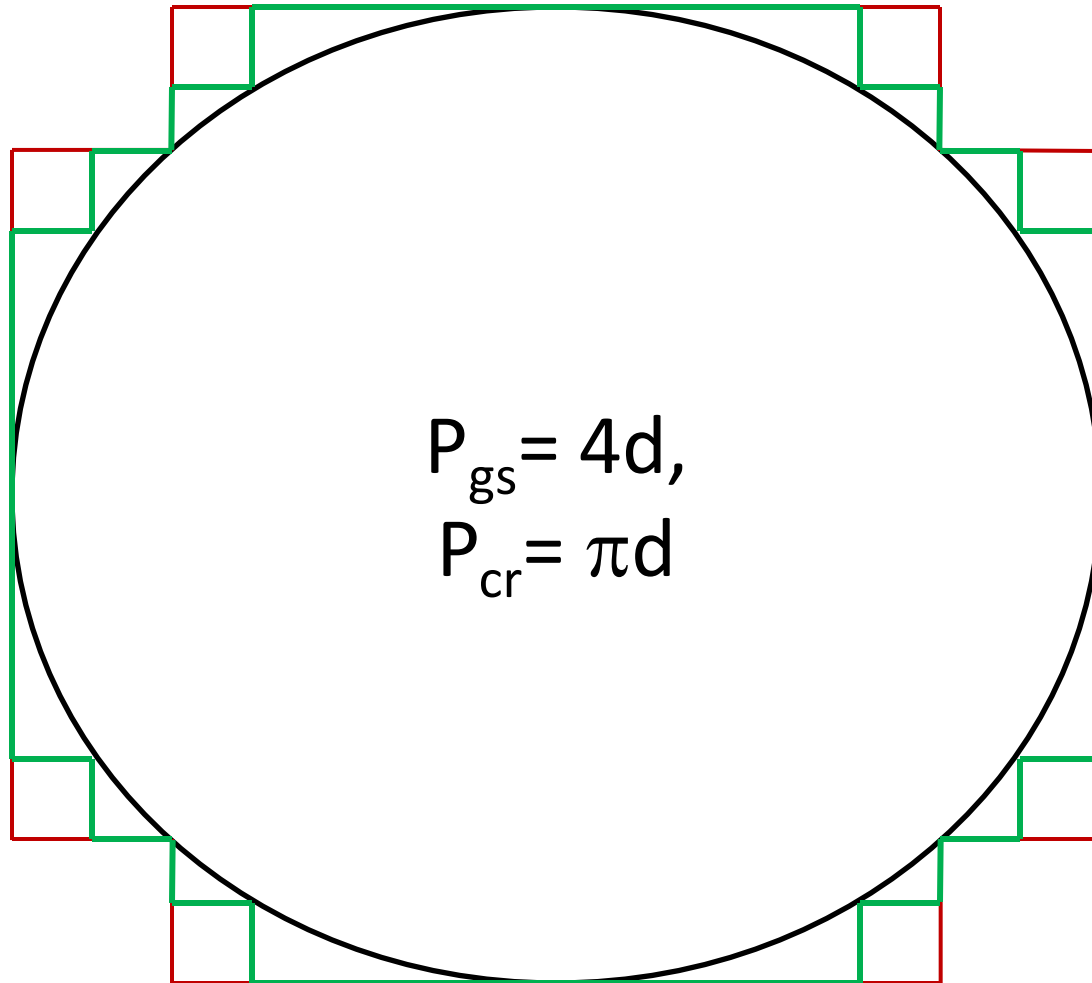
Proof



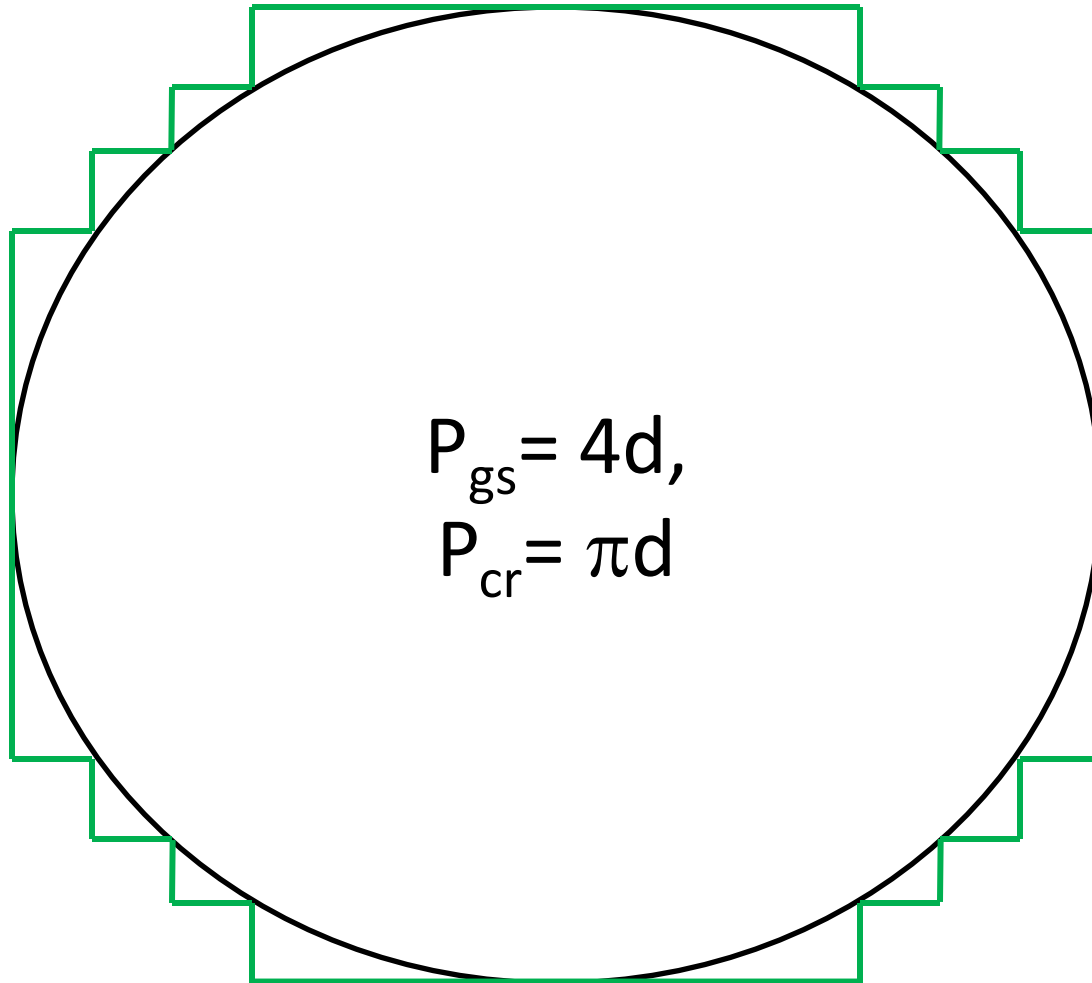
Proof (cont.)



Proof (cont.)



Proof (cont.)



Proof (cont.)

- The figure around the circle converges to the circle; hence its perimeter converges to πd .
- but the value of the perimeter is constant $4d$;
- hence $\pi=4$.

If you aren't convinced, then Poetry should help...

π is 4, – I don't joke!

4 is π , – I don't lie...

Draw a square near circle

(with diameter 1),

Cut its corners,
then new corners,

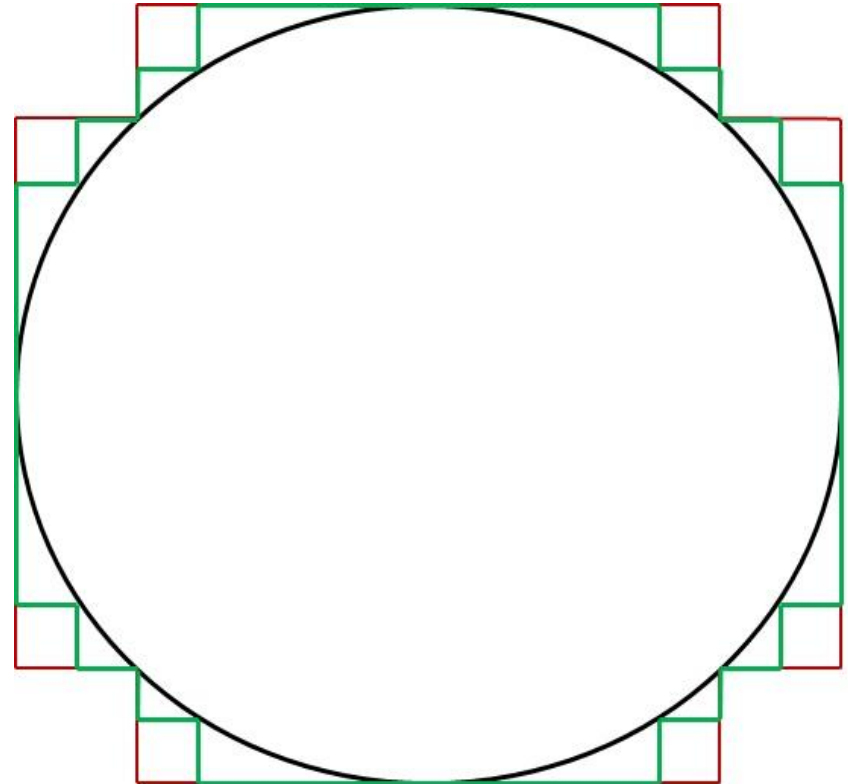
Proceed further
one by one.

4 is length of figure's border,

Length of circle equals π ;

Border line converges to circle,

It implies that 4 is π !



Formal Methods as a Rescue

- Let us specify the program in Hoare style by pre- and post-conditions.
- The pre-condition may be TRUE since the program has no input.
- The post-condition should be $pi_val==4.0$ due to exercises of the program.
- So we may hope to prove the following total correctness assertion

$$\models [TRUE] PiMC [pi_val=4.0].$$

Formal Methods as a Rescue

- But if we try to apply *axiomatic semantics* to generate verification conditions and prove the assertion then we encounter a problem of axiomatic semantics of the assignment

```
r = rand() % 100000000;
```

that has 2 instances in the program.

Part III

TYPES OF FORMAL SEMANTICS FOR FORMAL LANGUAGES


Syntax, Semantics, Pragmatics

- Programming Language is any artificial language designed to organize data processing.
- Every language (artificial or natural) may be characterized by its *syntax*, *semantics*, and *pragmatics*.

Syntax, Semantics, Pragmatics

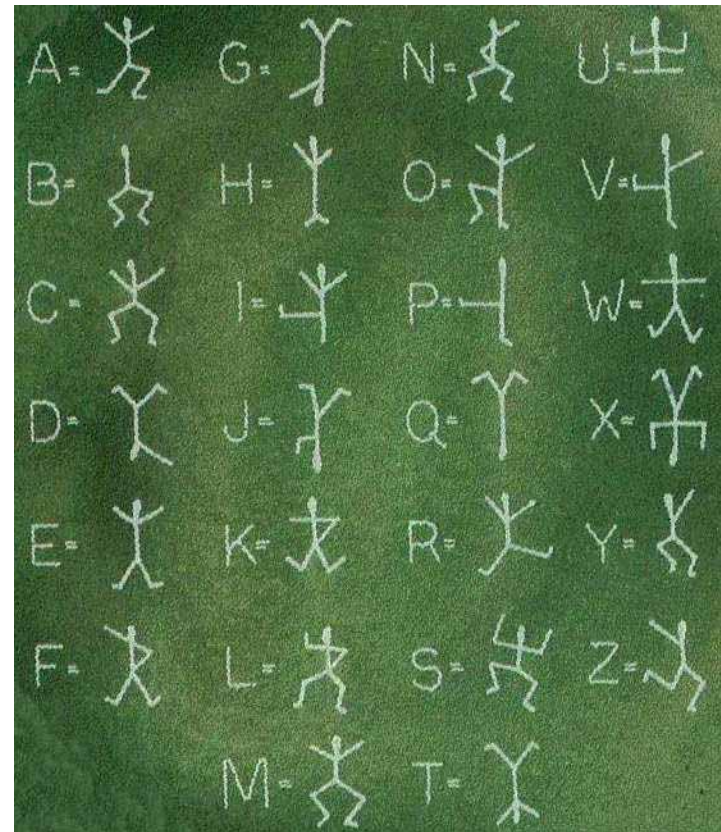
- *Syntax* is *orthography* of the language, rules to write correctly.
- *Semantics* is about methods to assign meaning to syntactically correct writings.
- *Pragmatics* is about use of the syntactically correct meaningful writings.

The Adventure of the Dancing Men

- One of the 56 *Sherlock Holmes* short stories written by Arthur Conan Doyle.
- Mr. Hilton Cubitt gives Sherlock Holmes a piece of paper with this mysterious sequence of stick figures: 
- These dancing men are at the heart of a mystery which seems to be driving his young wife Elsie to distraction.

The Adventure of the Dancing Men

Holmes realizes that it is a substitution cipher. He cracks the code by frequency analysis.



The Adventure of the Dancing Men

- Syntax is just as plain English with symbols instead of letters.
- Semantics is provided by transformation to plain English.
- Pragmatics: a cryptosystem of Chicago gangsters.

Esoteric Programming Languages

- *An esoteric programming language (esolang)* is a programming language designed to test the boundaries of computer programming language design
 - as a proof of concept,
 - or as a joke.

Esoteric Programming Languages

- The use of esoteric distinguishes these languages from programming languages that working developers use to write software.
- Usually, an esolang's creators do not intend the language to be used for mainstream programming.

Toy Esoteric Language TEL

- TEL is not a *programming* language at all, it is not designed for data processing.
- Its *pragmatics* is to introduce and explain different types of formal semantics:
 - Operational,
 - Denotational,
 - Axiomatic,
 - Second-order.

TEL *informal* syntax

- TEL sentences just look like structured programs, e.g.:

```
if z < 0 then z := -1
else (x := 0 ; y := 0 ;
while y ≤ z do
(y := y + 2 * x + 1 ;
x := x + 1) ;
x := x - 1) .
```

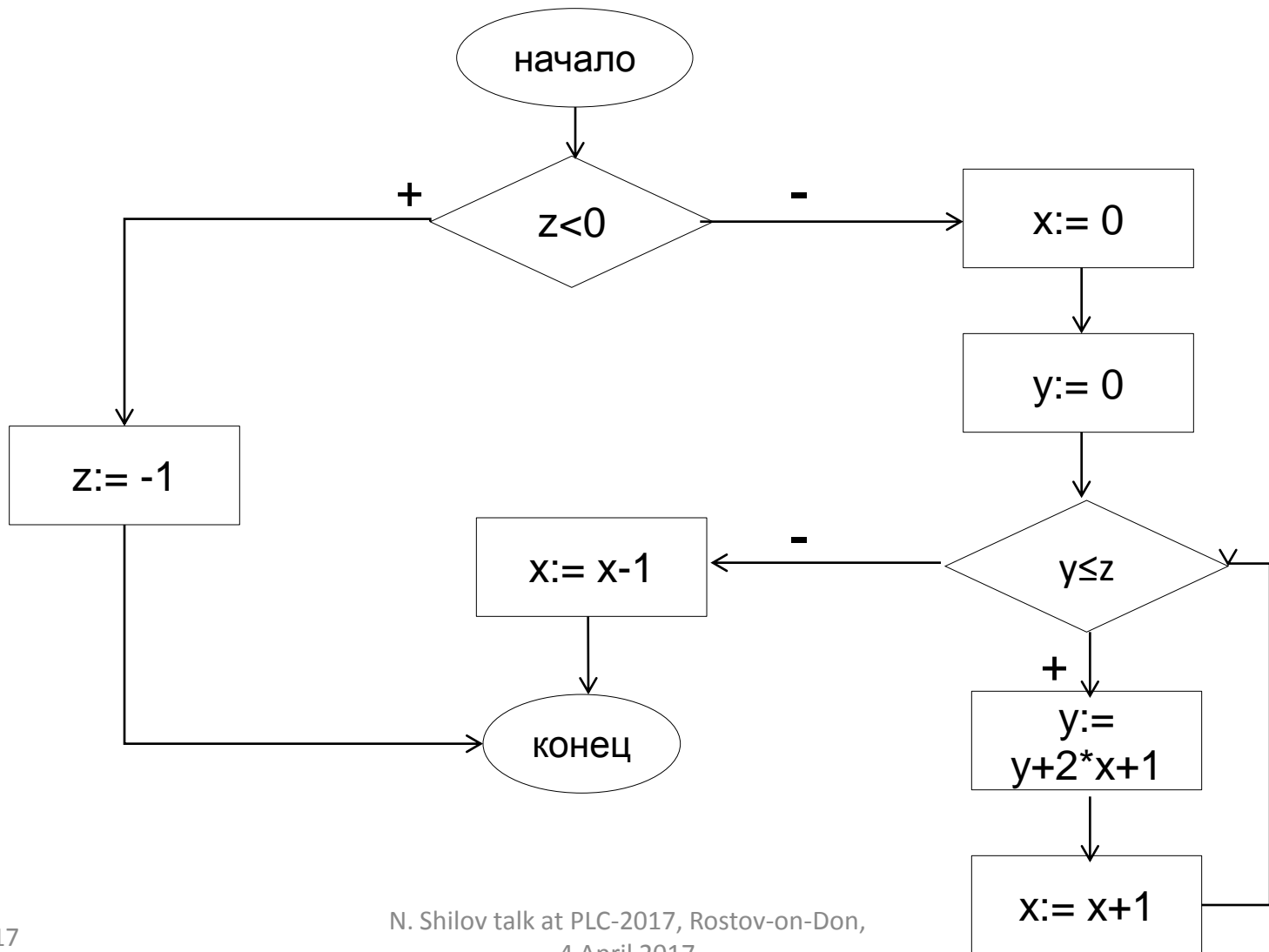
TEL *informal* syntax

- Correct TEL sentences are “programs” constructed from assignments by means of
 - compound “;”,
 - choice “if-then-else”,
 - loop “while-do”constructs.

TEL *informal* semantics

- Since every correct TEL sentence looks like an iterative program, one can draw a flowchart of this program.
- Every flowchart is a graph with assignments and conditions as nodes and control passing as edges.

TEL informal semantics: example

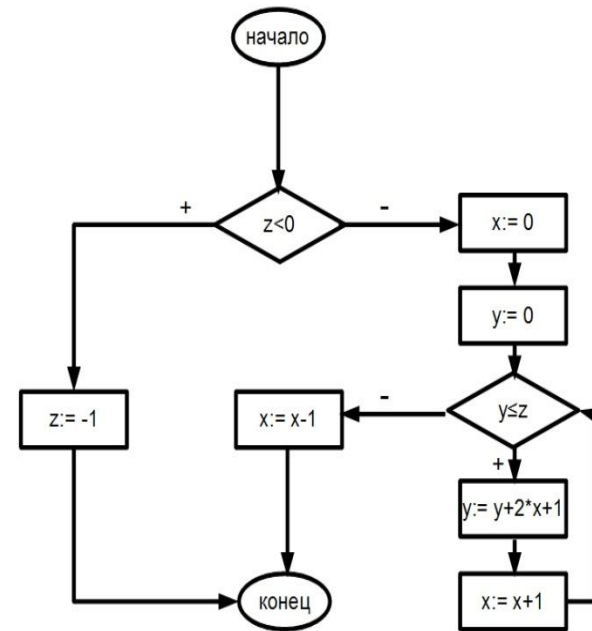


TEL *informal* semantics

- Let us count length of a path between nodes in a flowchart by number of assignments in this path (i.e. we do not count conditions at all).
- Then let semantics of a correct TEL sentence be the shortest length of a path through the corresponding flowchart (i.e. from start to finish).

TEL informal semantics: example

Semantics of the sample sentence is 1.



Operational Semantics: executable state machines

- *Operational semantics* translates programs to corresponding state machines (“mechanical procedures” of a certain class) whose “operations” (that change machine’s state) are (conventionally) “executable”: semantics of a program is defined in terms and by means of all admissible “executions” (i.e. runs) of the corresponding machine.

TEL operational semantics

- In the case of TEL, the target class of machinery consists of arithmetic expressions that are constructed from natural numbers (including 0 and 1) by means of addition and minimization operations.

TEL operational semantics

- The translation is defined as follows:
 - $F(x:=t) = 1$ for assignment;
 - $F(\beta;\gamma) = F(\beta) + F(\gamma)$;
 - $F(\text{if } \zeta \text{ then } \beta \text{ else } \gamma) = \min\{F(\beta), F(\gamma)\}$;
 - $F(\text{while } \zeta \text{ do } \beta) = 0$.

Denotational Semantics: an algebra for calculations

- Algebra is a set of objects with operations on/with them.
- Natural numbers \mathbf{N} with constants 0 and 1, binary operations “+” and “-” is an example of algebra.
- The same domain \mathbf{N} with constant 0, unary operation “+1” and binary operation “min” is another algebra (due to different operations).

Denotational Semantics: an algebra for calculations

- *Denotational semantics* assigns (in a consistent compositional compatible manner) the
 - elements of some algebra to correct sentences,
 - and the operations of this algebra to sentence constructs.

Denotational Semantics: an algebra for calculations

- Usually the assigning function is denoted by $[[\]]$.
- An element $[[\alpha]]$ that is assigned to a sentence α by $[[\]]$ is called *denotation* of/for α .
- An operation $[[\bullet]]$ that is assigned to a construct \bullet by $[[\]]$ is called *denotation* of/for \bullet .

TEL Denotational Semantics

- Let us fix natural numbers \mathbf{N} with constants 0 and 1 and binary operations “+” and “min”.
- Let $[[\]]$ be the following mapping:
 - $[[x:=t]] = 1$ for assignment;
 - $[[;]] = +$, $[[\text{if-then...else...}]] = \text{min}$, $[[\text{while-do...}]] = 0$;
 - $[[\text{constr}(\alpha, \beta)]] = [[\text{constr}]]([[\alpha]], [[\beta]])$ for any construct in “;”, “if-then...else...”, “while-do...”.

TEL semantics: operational vs. denotational

- **Proposition 1:** $val(F(\alpha)) = [[\alpha]]$ for every correct TEL sentence α .
- **Proof** by induction on syntax structure of α .
- So operational and denotational semantics of TEL match each other or are sound with respect to each other.

Axiomatic Semantics: code-driven proofs

- *Axiomatic system* is a calculus, i.e. a set of syntactic *inference rules* for deriving (“proving”) new “facts” (that are called *theorems*) from axioms (i.e. inference rules without premises).

TEL axiomatic semantics

- Axiomatic semantics for TEL is an axiomatic system for assertions of the following form

$$m \leq \alpha \leq n$$

where

- m is natural number,
- α is correct TEL sentence,
- n is a natural number such that $m \leq n$, or symbol « ∞ ».

TEL axiomatic semantics

Assignment axiom: $\frac{}{1 \leq \langle \text{assignment} \rangle \leq 1}$

Loop axiom: $\frac{}{0 \leq \text{while...do...} \leq 0}$

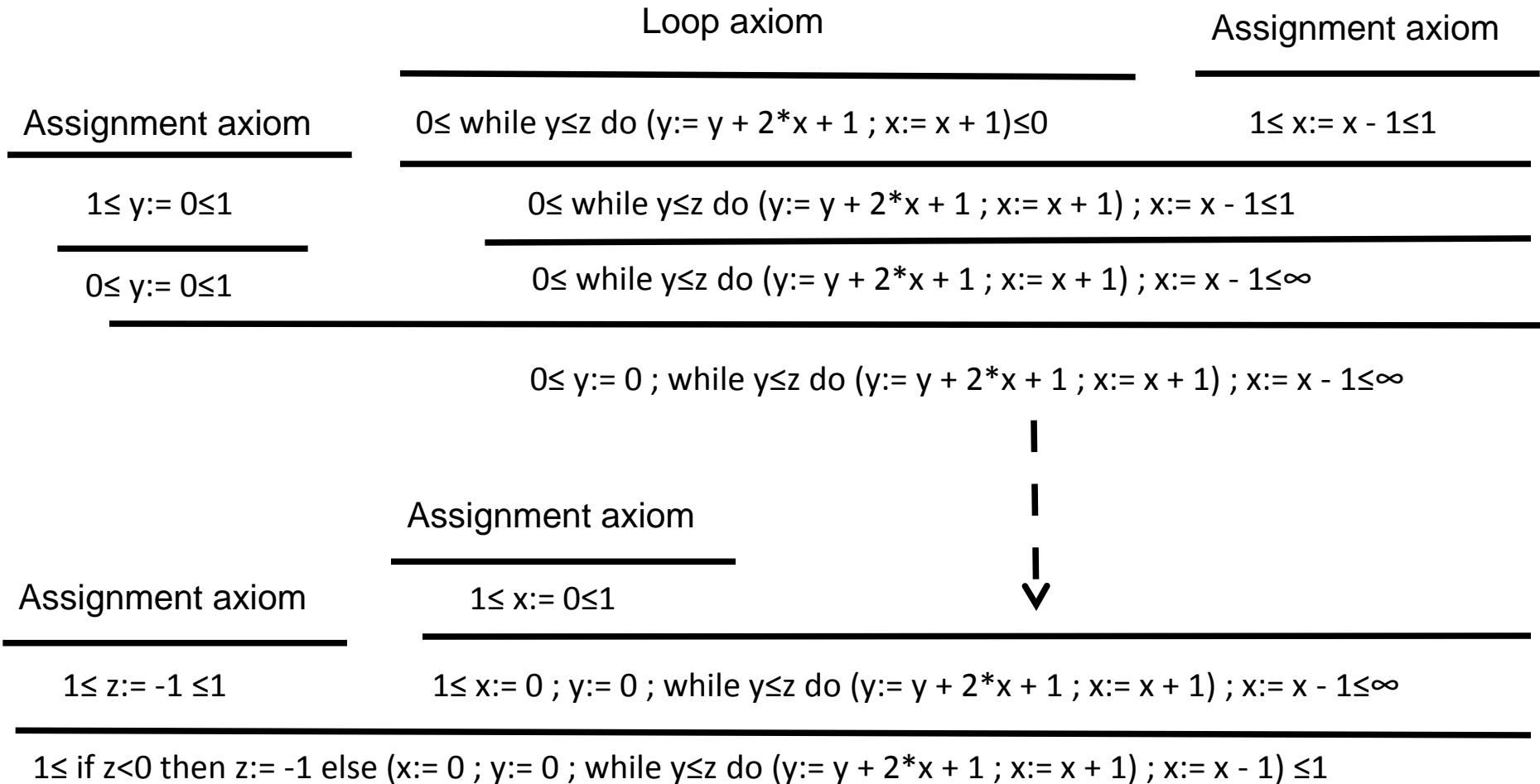
Block rule: $\frac{m \leq \alpha \leq n}{m \leq (\alpha) \leq n}$

Stretching rule: $\frac{m' \leq \alpha \leq n'}{m \leq \alpha \leq n}, m \leq m', n' \leq n$

Composition rule: $\frac{m' \leq \alpha \leq n' \quad m'' \leq \beta \leq n''}{m \leq \alpha; \beta \leq n}, m = m' + m'', n = n' + n''$

Then rule: $\frac{m \leq \alpha \leq n \quad m \leq \beta \leq \infty}{m \leq \text{if...then } \alpha \text{ else } \beta \leq n}$ Else rule: $\frac{m \leq \alpha \leq \infty \quad m \leq \beta \leq n}{m \leq \text{if...then } \alpha \text{ else } \beta \leq n}$

TEL axiomatic semantics: example



Validity vs. Provability, Soundness vs. Completeness

- Assertion $m \leq \alpha \leq n$ is said to be *valid*, if $m \leq [[\alpha]] \leq n$.
- Axiomatic semantics is said to be
 - *sound*, if all provable assertions are valid;
 - *complete*, if all valid assertions are provable.

Validity vs. Provability, Soundness vs. Completeness

- **Proposition 2:** *Tel axiomatic semantics is sound and complete.*
- **Proof:**
 - soundness – induction on height of the proof,
 - completeness – induction by sentence structure.

Part iV

A PUZZLE TO TEACH/LEARN FORMAL MODELS OF CONCURRENCY

Concurrency vs. Parallelism

By parallelism, I mean using extra computational resources to solve a problem faster. By concurrency, I mean correctly and efficiently managing access to shared resources. While using these terms in this way is not entirely standard, the distinction is paramount.

D. Grossman

Ready-For-Use: 3 Weeks of Parallelism and Concurrency
in a Required Second-Year Data-Structures Course.

Types of Formal Models of Concurrency

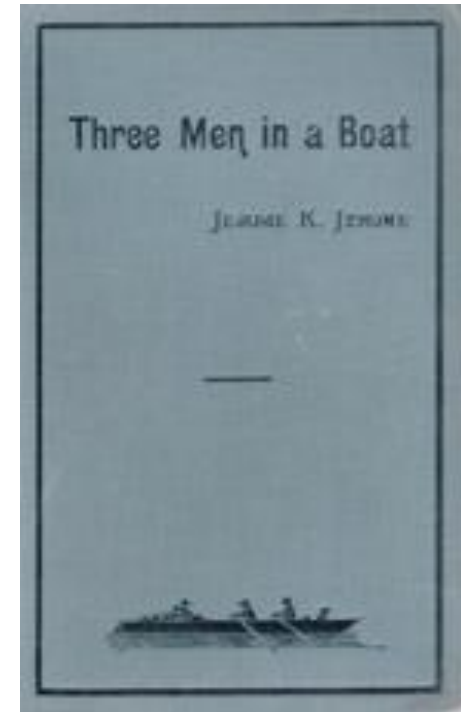
- *Petri nets* is a purely *semantic* model of parallelism.
- *Communicating Sequential Processes (CSP)* is an *algebraic formal language* with fixed syntax and *denotational semantics*.
- *Syntactic calculi* that formalize different aspects of parallelism: the *Calculus of Communicating Systems (CCS)*, the *Pi-Calculus* for communicating mobile systems, the *Ambient Calculus*, etc.

Types of Formal Models of Concurrency

- *Labeled Transition Systems* (LTS) naturally emerge in semantic, algebraic and syntax formal models.
- *Dynamic* and *temporal logic(s)* are used for specification and verification of (concurrent and) parallel systems, presented by LTS.

One Puzzle in Different Formalisms

Fascinating and fabulous puzzle *Four men and a Boat* is good to illustrate and compare 4 (at least) formal.



Four Men and a Boat Puzzle



Four men Albert, Conrad, Donald and Edmund are on the left bank of a river and need to move to the right bank by a boat that has 2 seats and one pair of oars.

Four Men and a Boat Puzzle

- Sporty Albert can cross the river by the boat without a companion in 5 minutes (in any direction, forth and back),
- regular Conrad can do the same in 10 minutes,
- fatty Donald – in 20 minutes, and
- fat Edmund – in 25 minutes.

Four Men and a Boat Puzzle

When any two men are crossing the river together the pace of the boat is defined by the fattest man in the pair, ex., Albert and Donald together can cross the river in 20 minutes.



Question: do these four men can cross the river in one hour?

Enjoy the Puzzle!

- This is a reachability (i.e. “simple”) but a very challenging puzzle! Typically 8 in 10 students (in my experience) first “prove” that the four men cannot cross the river in one hour.

Enjoy the Puzzle!

- They usually claim that it is “obvious” that sporty Albert have to accompany (convoy) other men because he is the fastest and it would be better him to transport the boat back every time; under this assumption transportation of 4 men takes 1 hour and 5 minutes.

Two ways to refute wrong belief

Human-oriented way comprises two steps:

- first someone must solve the puzzle (it needs some ingenuity),
- then prove manually impossibility of a solution where Albert convoys other men (that is very easy).

Two ways to refute wrong belief

A computer-aided approach:

- build the corresponding model labeled transition system (i.e. the reachability graph for the modeling Petri net, or reduction graph for the corresponding CCS process specification, etc.),

Two ways to refute wrong belief

- Formulate in a logic and check in the model the hypothesis that
 - *if a positive solution exists,*
 - *then there exists a solution where Albert convoys other men until all are on the right bank.*

Two ways to refute wrong belief

The hypothesis in CTL:

(Albert_at_Left & Conrad_at_Left &
& Donald_at_Left & Edmund_at_Left &
& Boat_at_Left & Timer_is_Set) →

→ (EF(Albert_at_Right & Conrad_at_Right &

& Donald_at_Right & Edmund_at_Right) →

→ E(Albert_on_Move U (Albert_at_Right &

& Conrad_at_Right &

& Donald_at_Right &

& Edmund_at_Right)

)

Thanks!

(Questions?)