Newtonian Program Analysis:

Solving Sharir and Pnueli's Equations

Javier Esparza

Technische Universität München

Joint work with

Stefan Kiefer and Michael Luttenberger

Two Approaches to Interprocedural Data Flow Analysis†

Micha Sharir
Amir Pnueli

7-1. INTRODUCTION

Under the general heading of program analysis we can find today two disciplines which, even though they have similar aims, differ in the means and tools they apply to the task of analysis. The first is the discipline of *program verification*. This is usually presented as the process of finding invariants of the program, or in other words fully characterizing the behavior of the program, discovering all the properties of all possible executions [Mann74, Cous77e]. As such, it is extremely ambitious and hence a priori doomed to failure on theoretical grounds for all but the most restricted program models.

Newtonian Program Analysis

JAVIER ESPARZA, STEFAN KIEFER and MICHAEL LUTTENBERGER
Technische Universität München

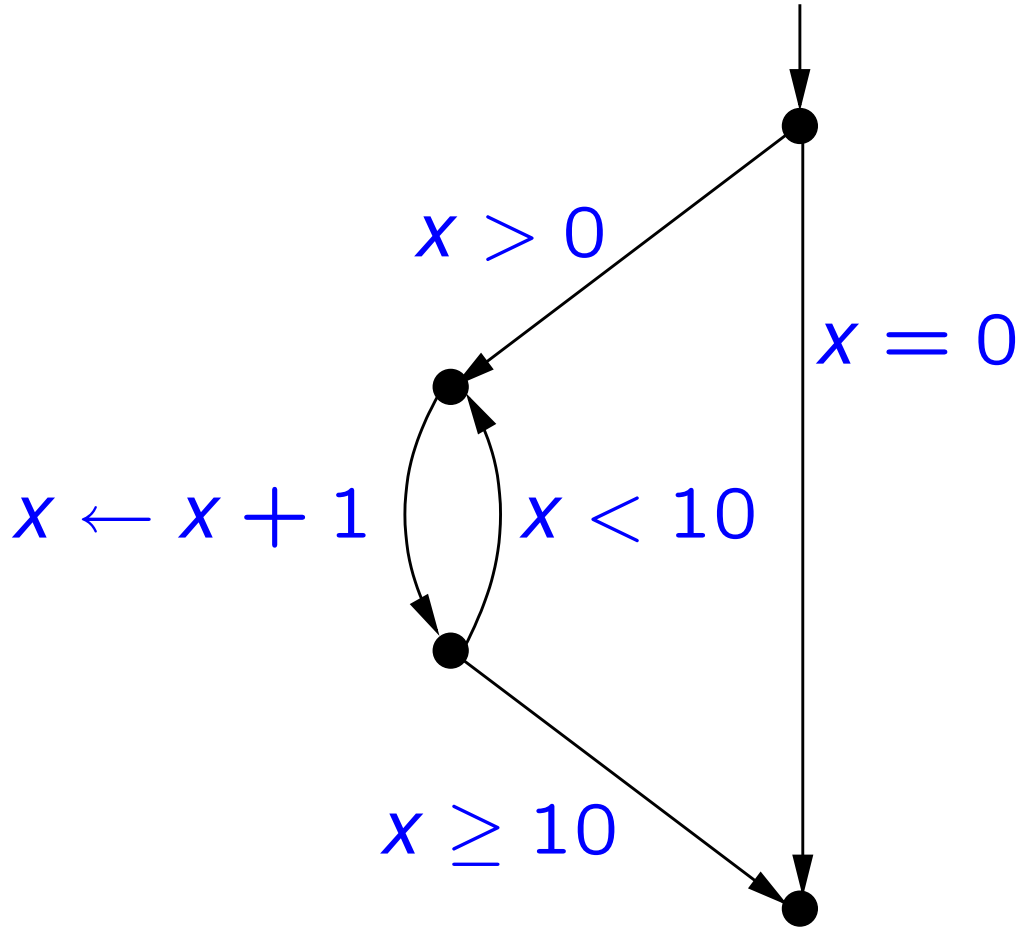
. . .

1. INTRODUCTION

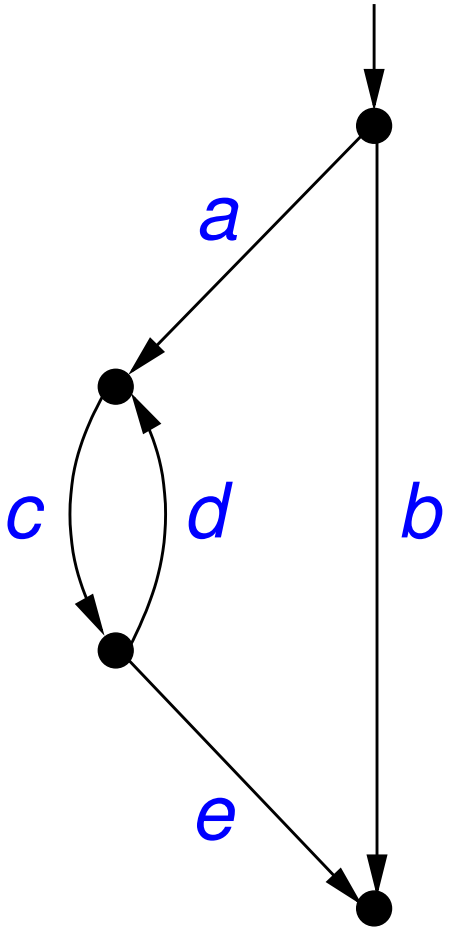
This paper presents a novel generic technique for solving dataflow equations in interprocedural dataflow analysis. It is obtained by generalizing Newton's method, the 300-year-old technique for computing a zero of a differentiable function.

Our approach to interprocedural analysis is very similar to Sharir and Pnueli's functional approach [Sharir and Pnueli 1981; Jones and Muchnick 1982; Knoop and Steffen 1992; Reps et al. 1995; Sagiv et al. 1996; Nielson et al. 1999; Reps et al. 2005]. Sharir and Pnueli assume the following as given: a (join-) semilattice[1] of *values*, a mapping assigning to every program instruction a value, and a con-

From programs to equations: Intraprocedural



From programs to equations: Intraprocedural

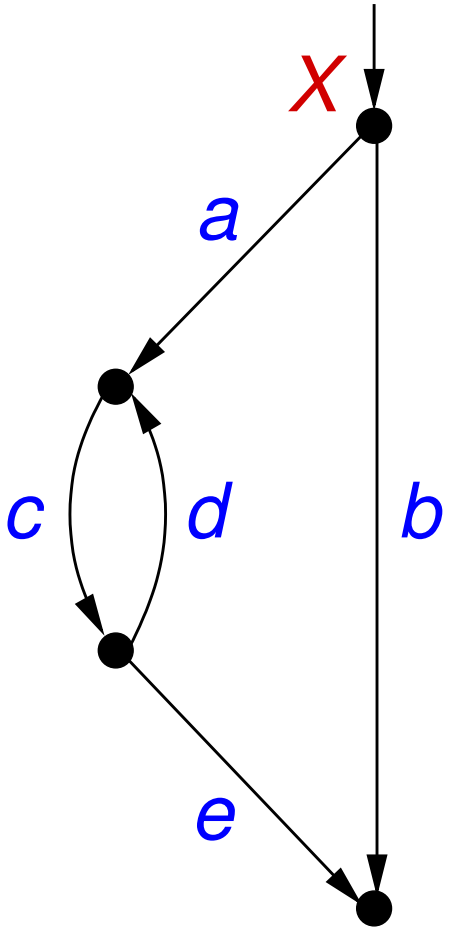


One-step relations

$a, \ldots, e \subseteq (\mathbb{N} \times \mathbb{N})$

$c = \{(x, x+1) \mid x \geq 0\}$

From programs to equations: Intraprocedural



One-step relations

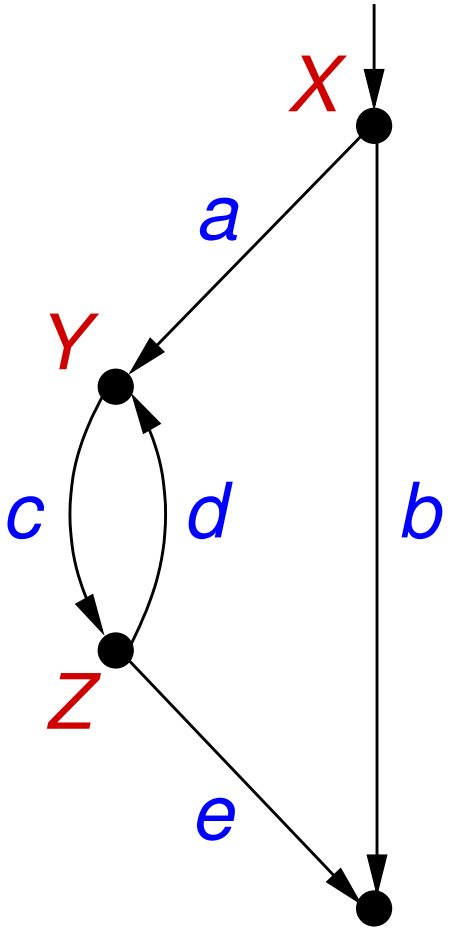
$a, \ldots, e \subseteq (\mathbb{N} \times \mathbb{N})$

$c = \{(x, x+1) \mid x \geq 0\}$

Big-step relation

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

From programs to equations: Intraprocedural



One-step relations

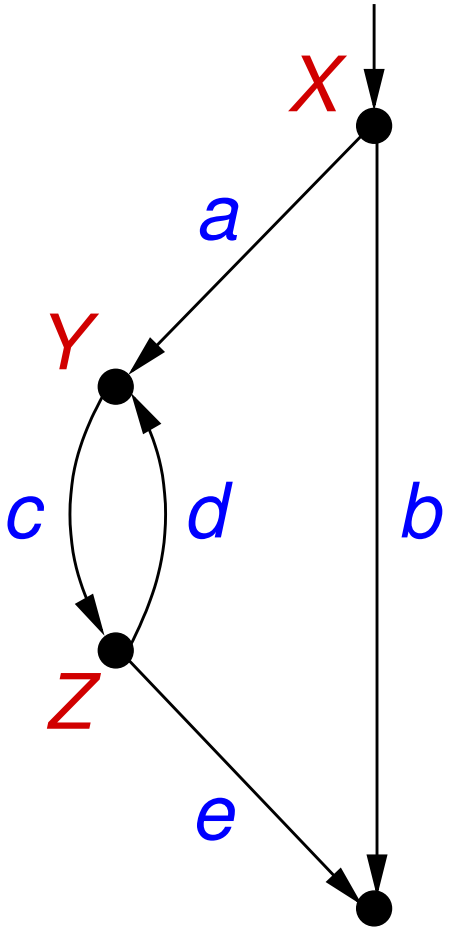
$a, \ldots, e \subseteq (\mathbb{N} \times \mathbb{N})$

$c = \{(x, x+1) \mid x \geq 0\}$

Big-step relations

$X, Y, Z \subseteq \mathbb{N} \times \mathbb{N}$

From programs to equations: Intraprocedural



One-step relations

$a, \ldots, e \subseteq (\mathbb{N} \times \mathbb{N})$

$c = \{(x, x+1) \mid x \geq 0\}$

Big-step relations

$X, Y, Z \subseteq \mathbb{N} \times \mathbb{N}$

$$X = a \cdot Y + b$$

$$Y = c \cdot Z$$

$$Z = d \cdot Y + e$$

From programs to equations: Intraprocedural

Program $\longmapsto$ system $X = f(X)$ of linear fixed-point equations

From programs to equations: Intraprocedural

Program $\longmapsto$ system $X = f(X)$ of linear fixed-point equations

Least solution non-computable in general

From programs to equations: Intraprocedural

Program $\longmapsto$ system $X = f(X)$ of linear fixed-point equations

Least solution non-computable in general

Program analysis: domain $2^{\mathbf{N}}$ $\longmapsto$ abstract domain D

transformer f $\longmapsto$ abstract transformer $f^{\#}$

From programs to equations: Intraprocedural

Program $\longmapsto$ system $X = f(X)$ of linear fixed-point equations

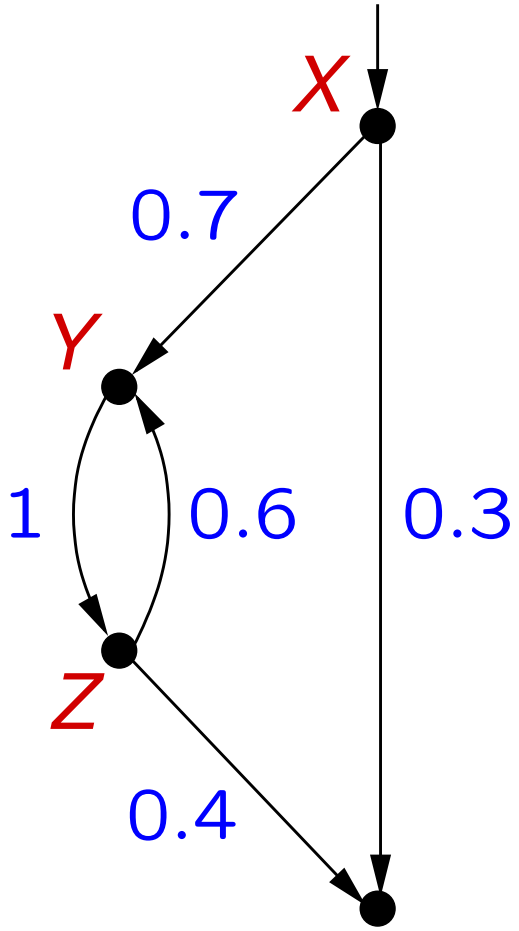
Least solution non-computable in general

Program analysis: domain $2^{\mathbf{N}}$ $\longmapsto$ abstract domain D

transformer f $\longmapsto$ abstract transformer $f^{\#}$

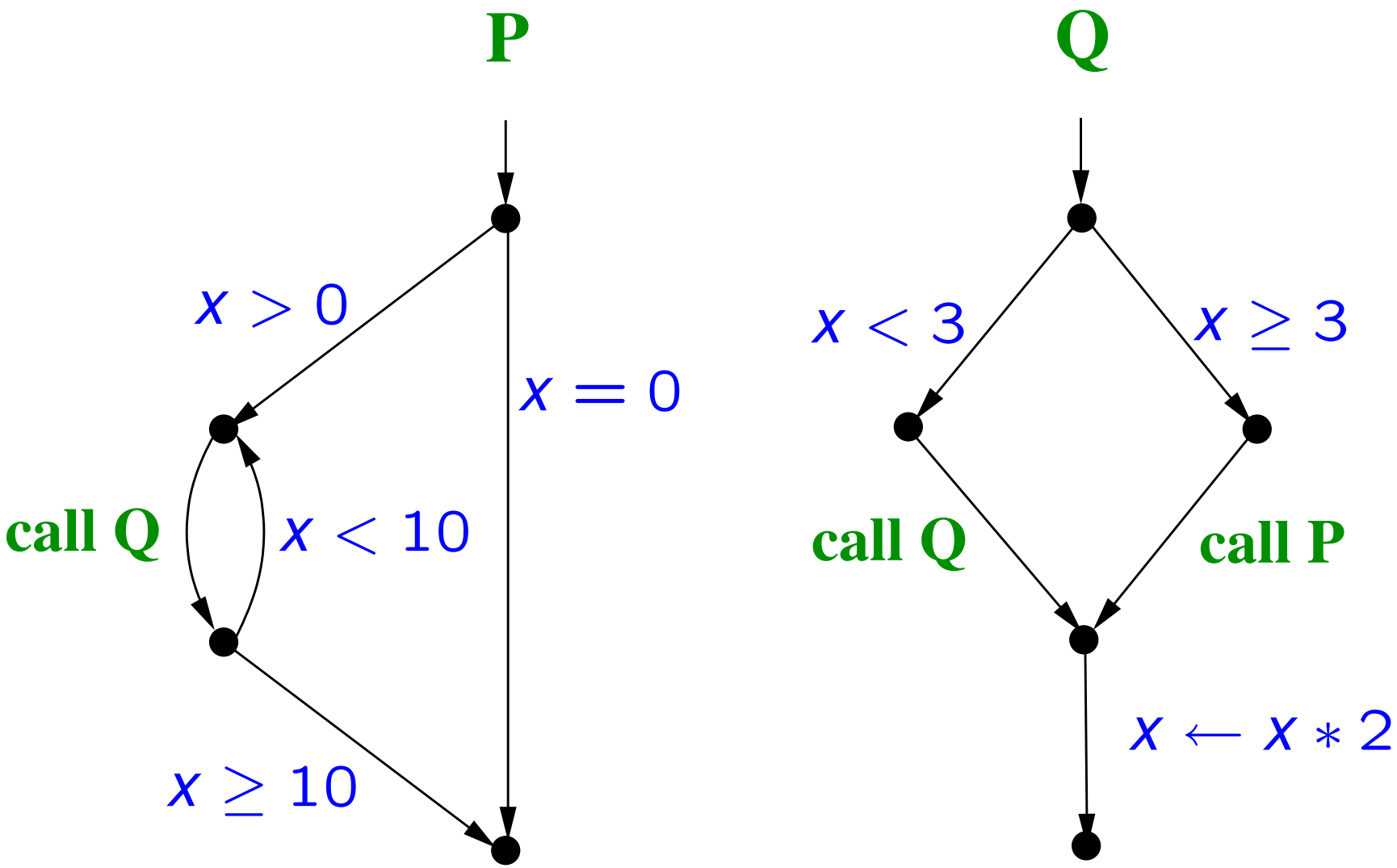
Sufficient condition for existence of least solution: $(D, +, \cdot)$ is a (ω-continuous) semiring

Quantitative program analysis: Expected time

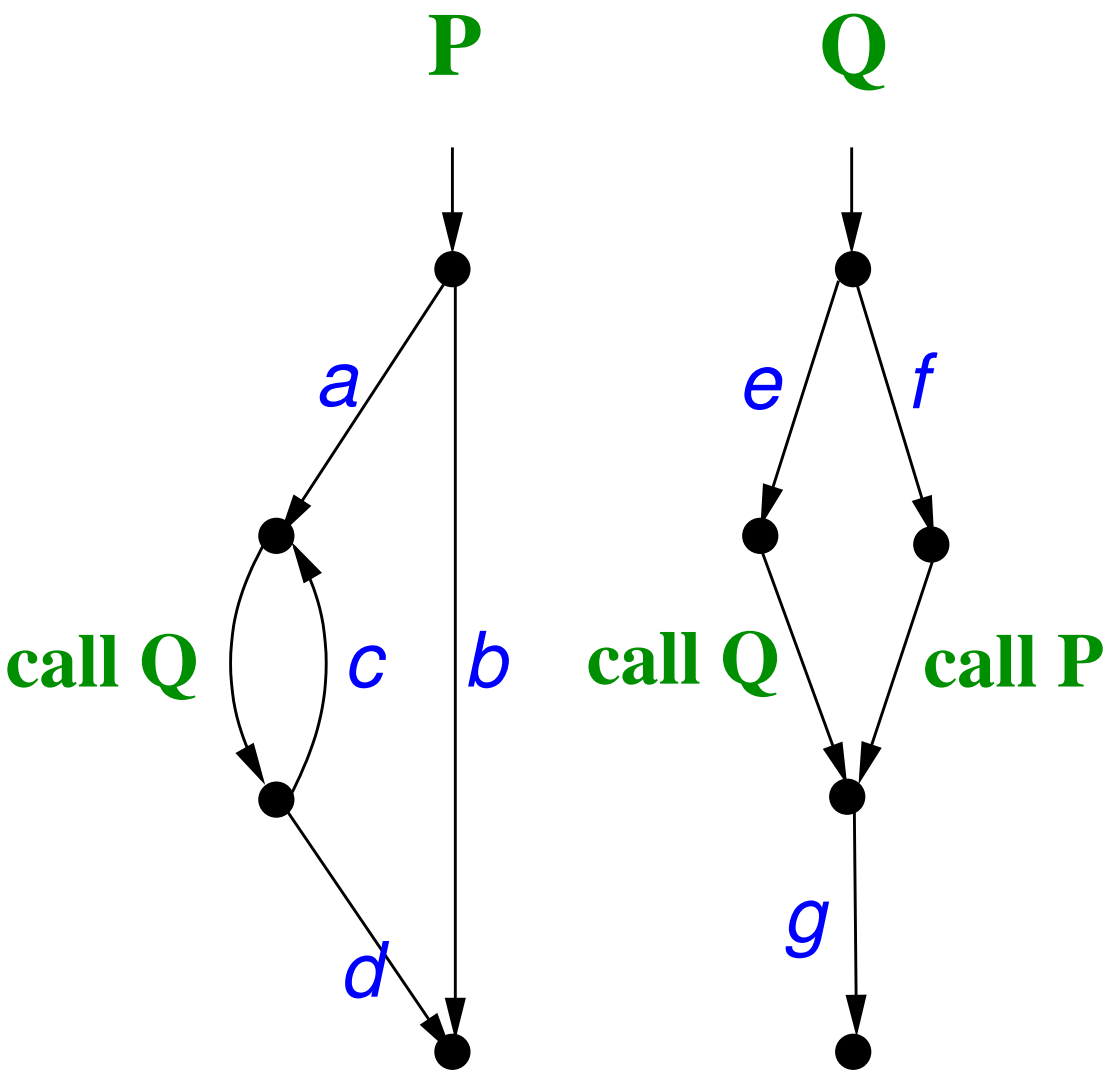


$$X = 0.7 \cdot Y + 1$$
$$Y = Z + 1$$
$$Z = 0.6 \cdot Y + 1$$

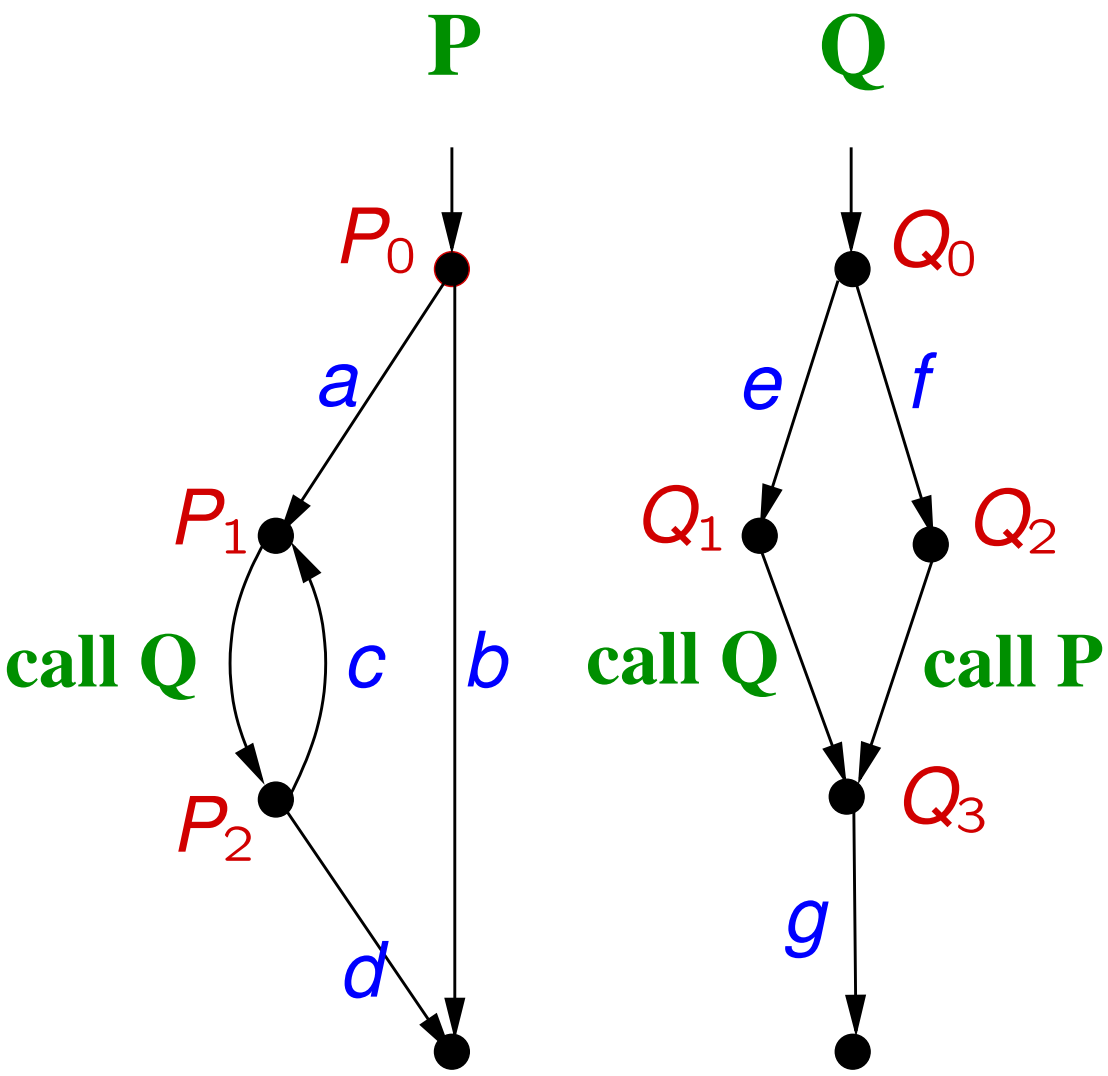
From programs to equations: Interprocedural



From programs to equations: Interprocedural



From programs to equations: Interprocedural



$$P_0 = a \cdot P_1 + b$$

$$P_1 = ?? \cdot P_2$$

$$P_2 = c \cdot P_1 + d$$

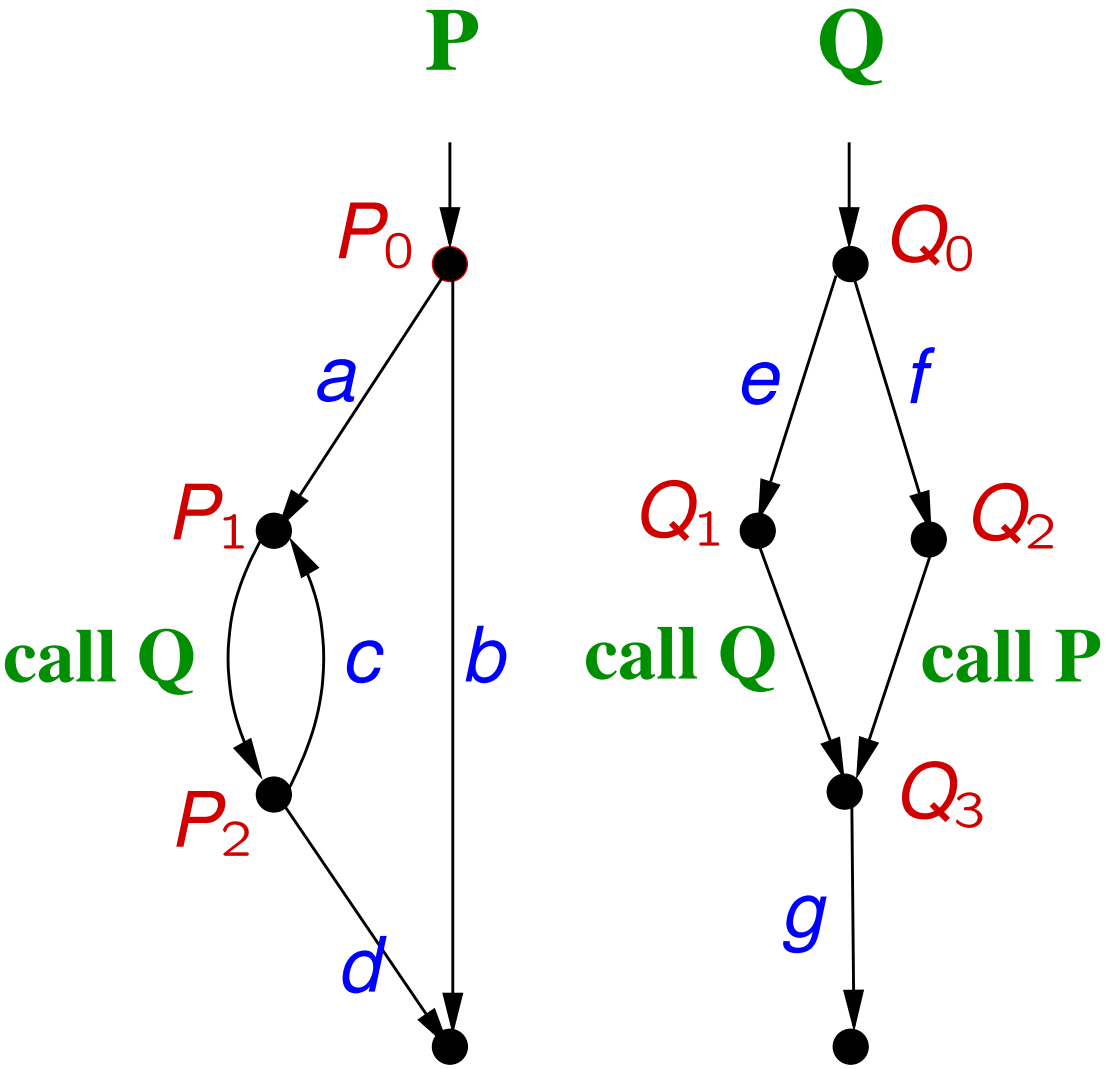
$$Q_0 = e \cdot Q_1 + f \cdot Q_2$$

$$Q_1 = ?? \cdot Q_3$$

$$Q_2 = ?? \cdot Q_3$$

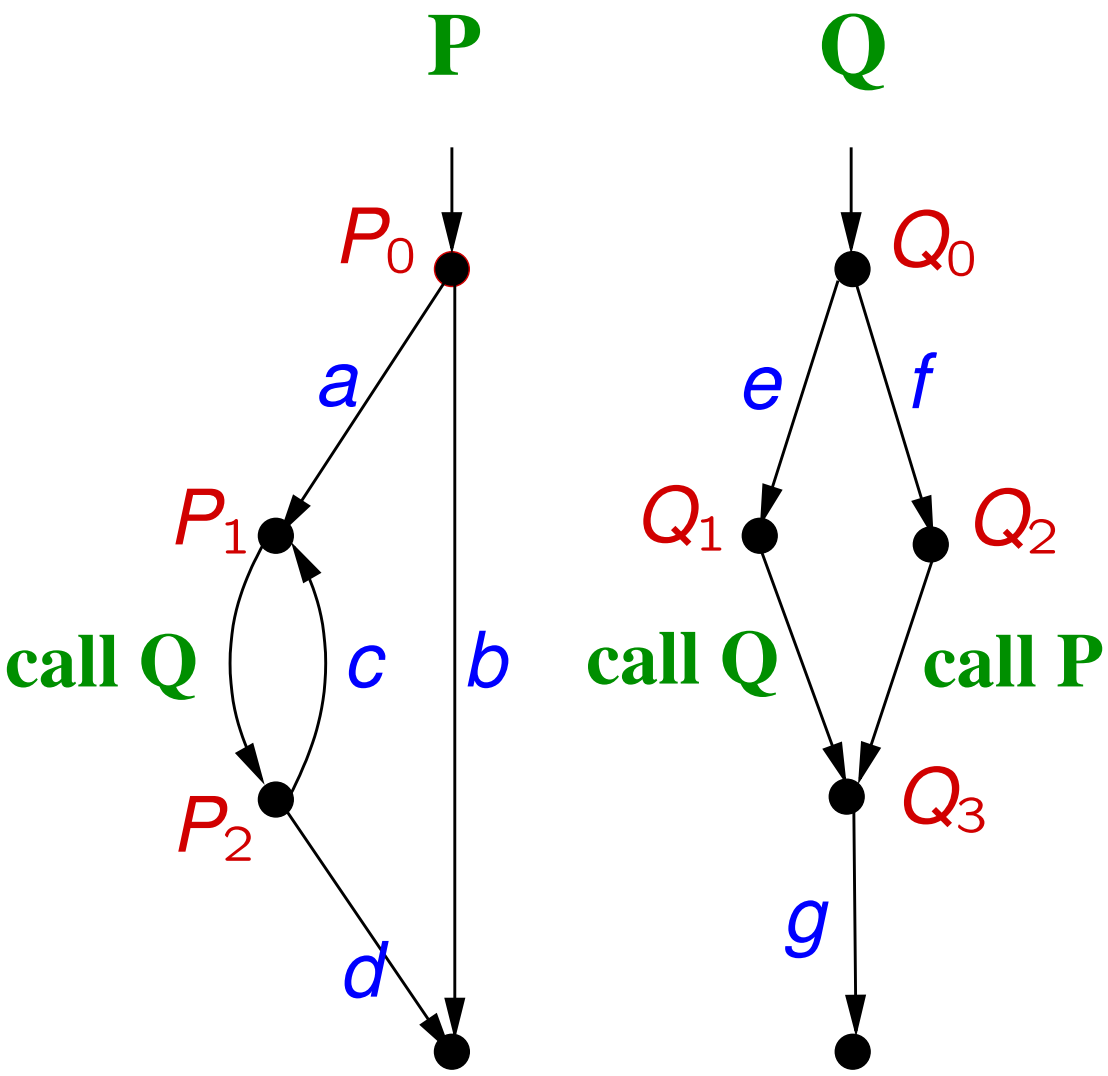
$$Q_3 = g$$

Sharir and Pnueli's functional approach



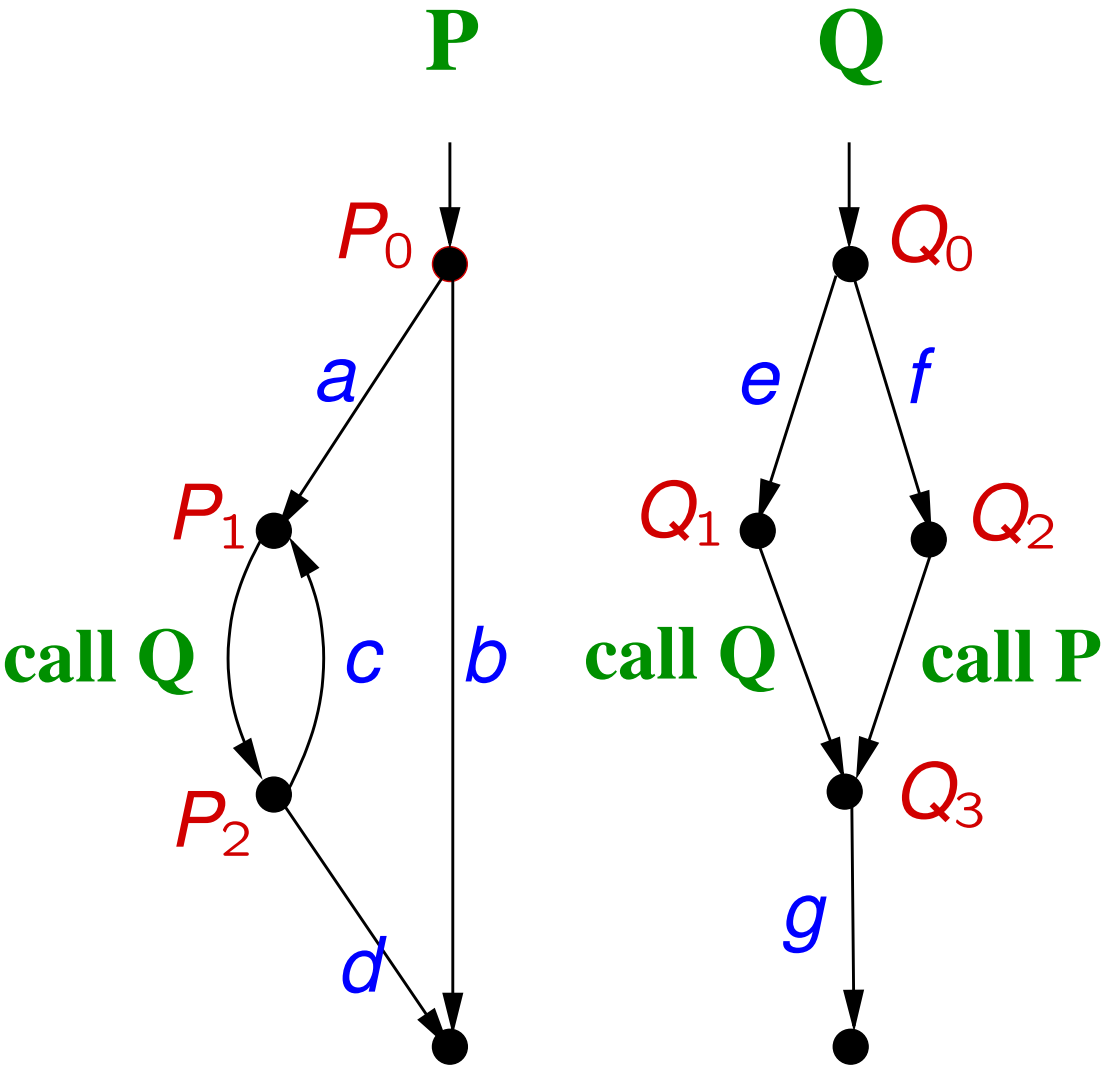
$$P_0 = a \cdot P_1 + b$$
$$P_1 = Q_0 \cdot P_2$$
$$P_2 = c \cdot P_1 + d$$
$$Q_0 = e \cdot Q_1 + f \cdot Q_2$$
$$Q_1 = ?? \cdot Q_3$$
$$Q_2 = ?? \cdot Q_3$$
$$Q_3 = g$$

Sharir and Pnueli's functional approach



$$P_0 = a \cdot P_1 + b$$
$$P_1 = Q_0 \cdot P_2$$
$$P_2 = c \cdot P_1 + d$$
$$Q_0 = e \cdot Q_1 + f \cdot Q_2$$
$$Q_1 = Q_0 \cdot Q_3$$
$$Q_2 = ?? \cdot Q_3$$
$$Q_3 = g$$

Sharir and Pnueli's functional approach



$$P_0 = a \cdot P_1 + b$$
$$P_1 = Q_0 \cdot P_2$$
$$P_2 = c \cdot P_1 + d$$
$$Q_0 = e \cdot Q_1 + f \cdot Q_2$$
$$Q_1 = Q_0 \cdot Q_3$$
$$Q_2 = P_0 \cdot Q_3$$
$$Q_3 = g$$

Sharir and Pnueli's interprocedural equations

Program $\longmapsto$ system $X = f(X)$ of polynomial, non-linear fixed-point equations

Sharir and Pnueli's interprocedural equations

Program $\longmapsto$ system $X = f(X)$ of polynomial, non-linear fixed-point equations

Least solution non-computable in general

Program analysis: domain $2^{\mathbf{N}}$ $\longmapsto$ abstract domain D

transformer f $\longmapsto$ abstract transformer $f^{\#}$

Sufficient condition for existence of least solution: $(D, +, \cdot)$ is a (ω-continuous) semiring

Solving the equations: Kleene iteration

Theorem [Kleene]: The least solution μf of $X = f(X)$ is the supremum of $\{k_i\}_{i \geq 0}$, where

$$\begin{aligned} k_0 &= f(0) \\ k_{i+1} &= f(k_i) \end{aligned}$$

Basic algorithm: compute k_0, k_1, k_2, ... until either $k_i = k_{i+1}$, which implies $k_i = \mu f$, or the approximation is considered adequate.

Kleene iteration is slow

Set domains: Kleene iteration never terminates for $X = f(X)$ if least solution μf is an infinite set.

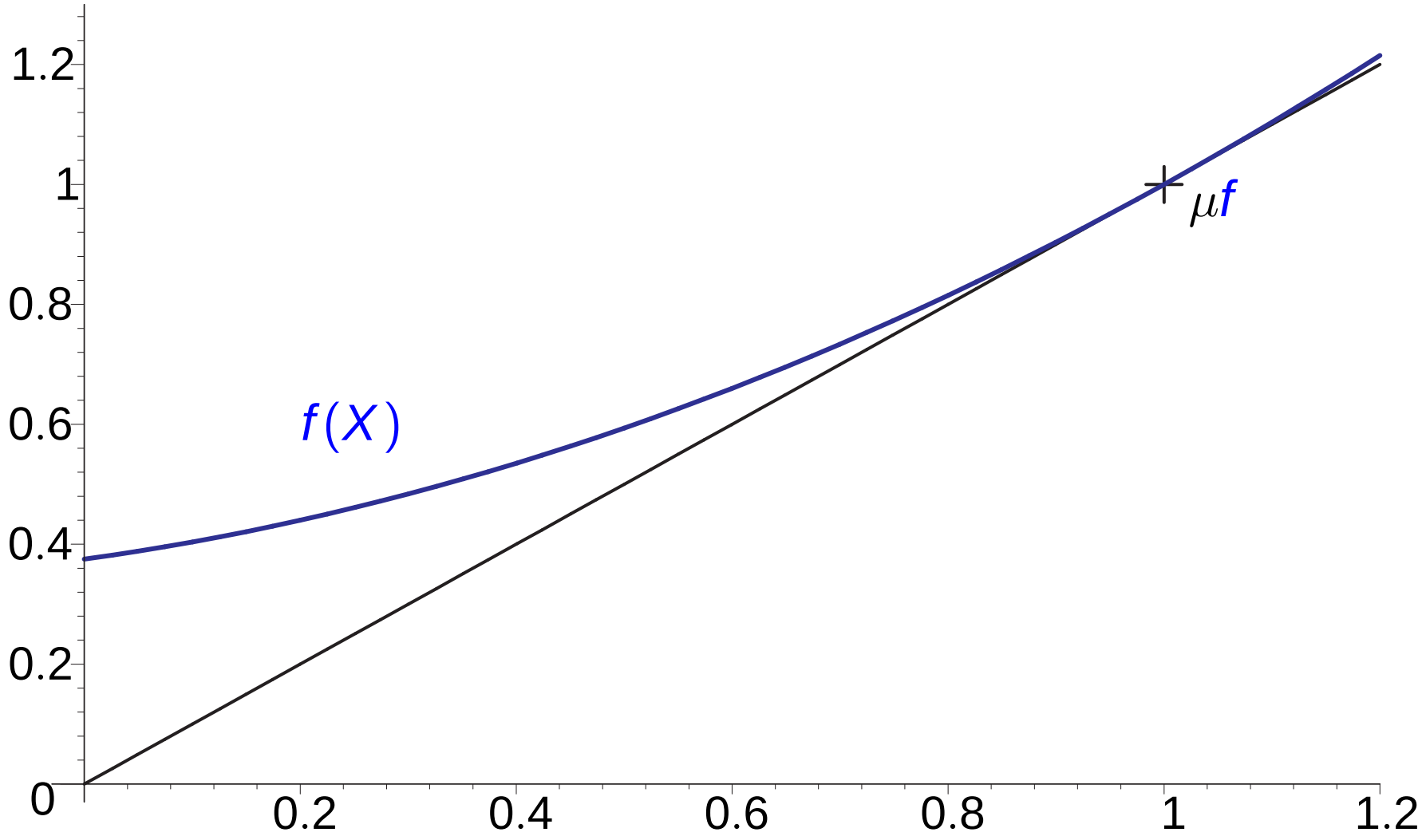
- $X = a \cdot X + b \qquad \mu f = a^* b$
- Kleene approximants are finite sets: $k_i = (\epsilon + a + \ldots + a^i)b$

Probabilistic interpretation: convergence can be very slow for polynomial equations [EY STACS05].

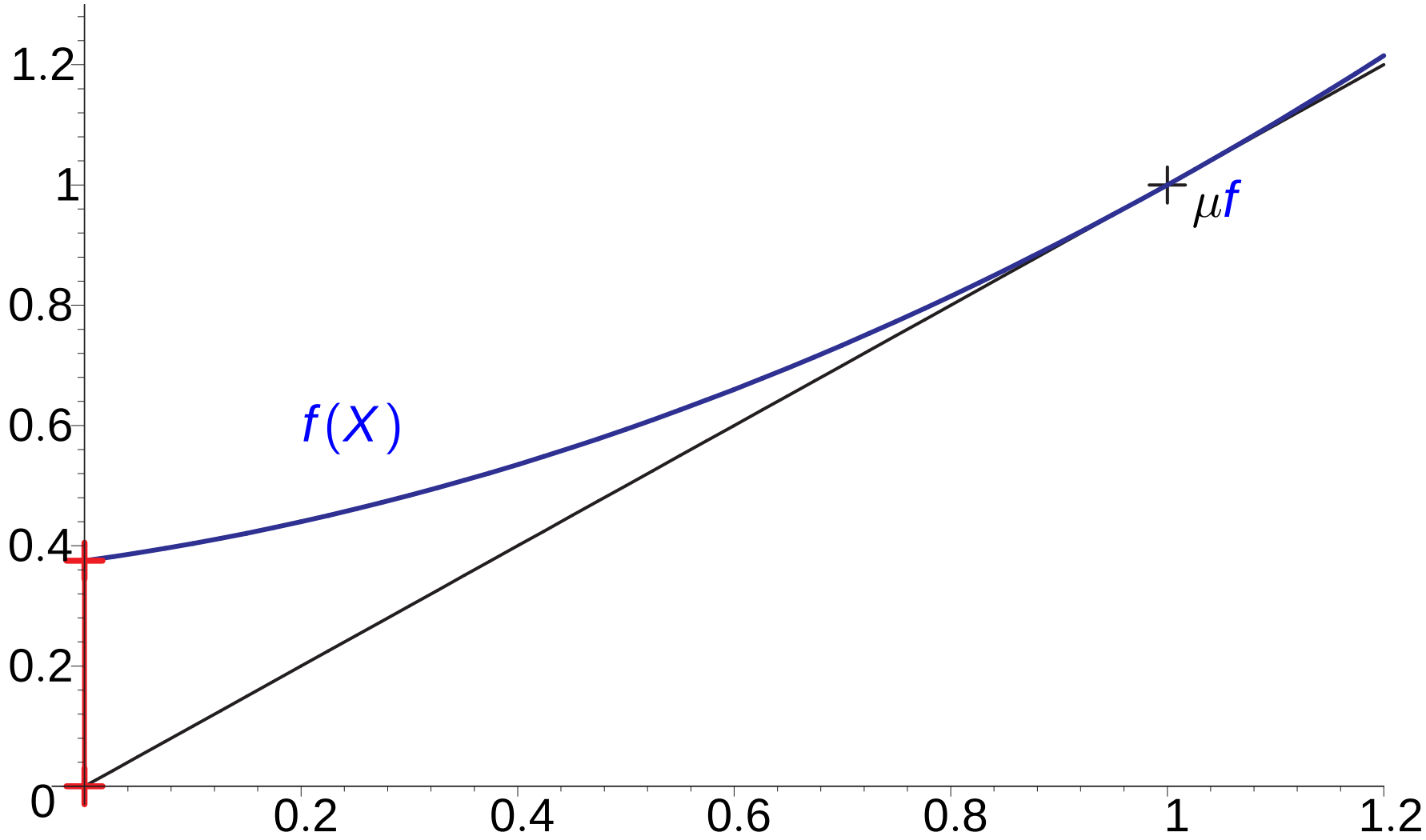
- $X = \frac{1}{2} X^2 + \frac{1}{2} \qquad \mu f = 1 = 0.99999\ldots$
- "Logarithmic convergence": k iterations to get $\log k$ bits of accuracy.

$$k_n \leq 1 - \frac{1}{n+1} \qquad k_{2000} = 0.9990$$

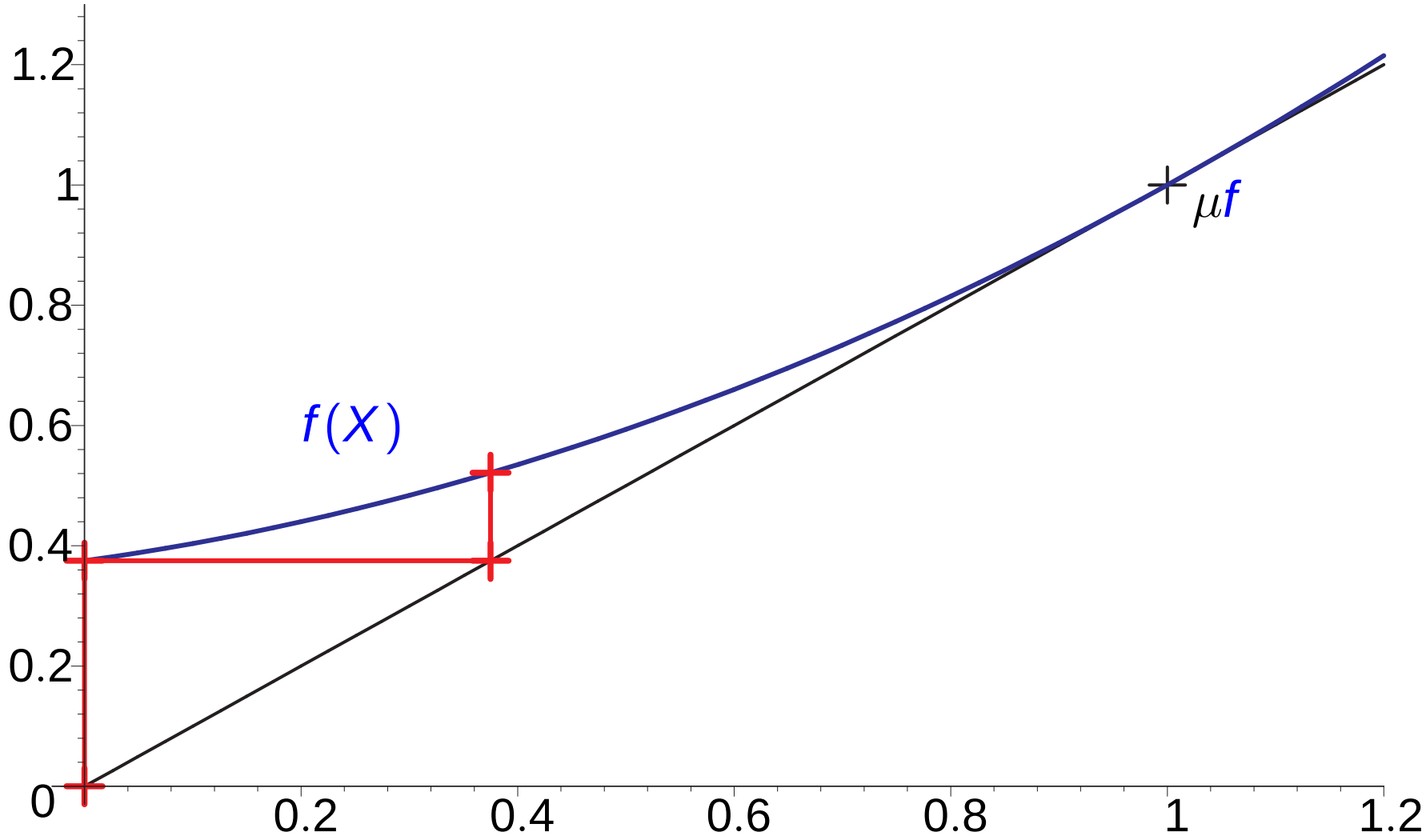
Kleene Iteration for $X = f(X)$ (univariate case)



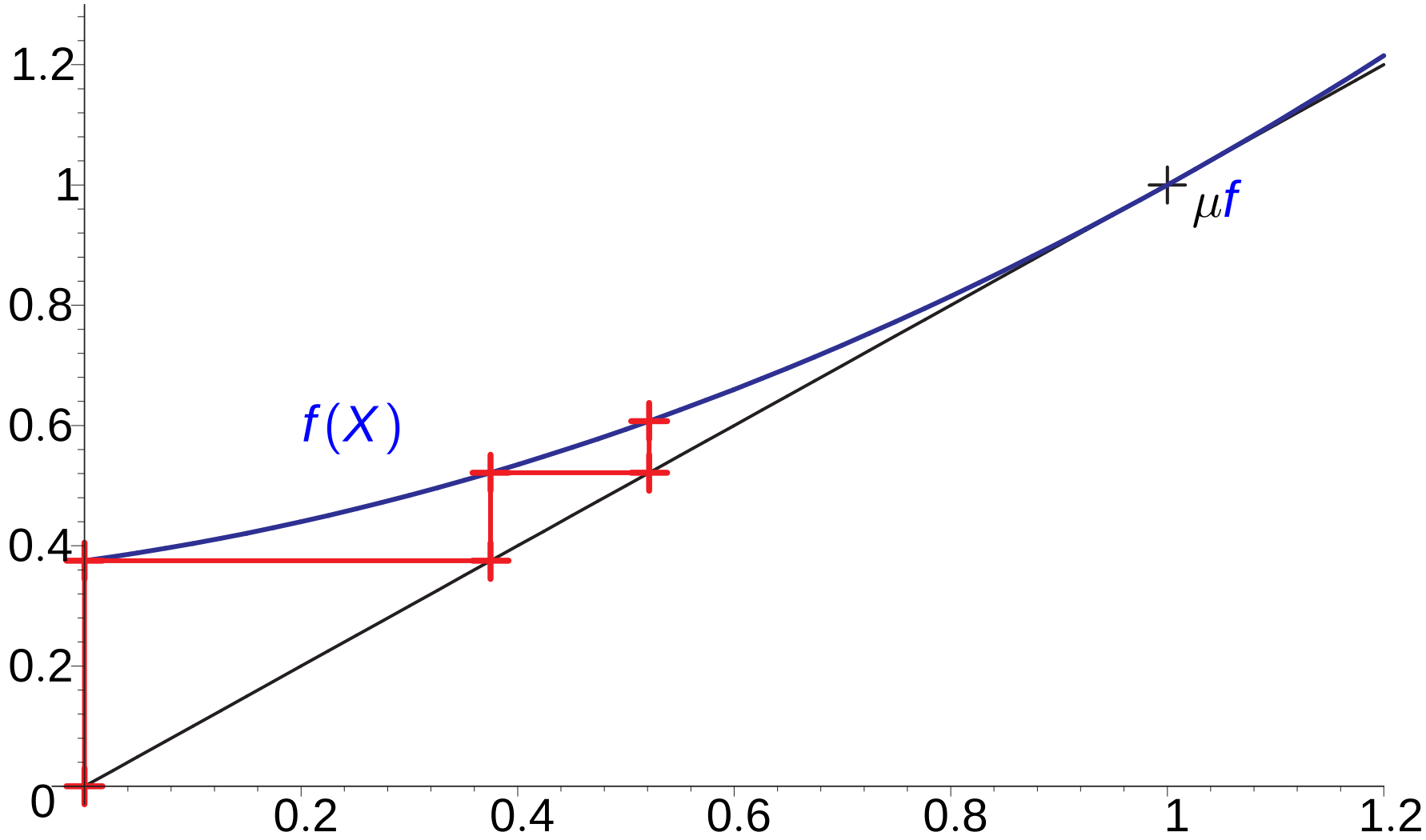
Kleene Iteration for $X = f(X)$ (univariate case)



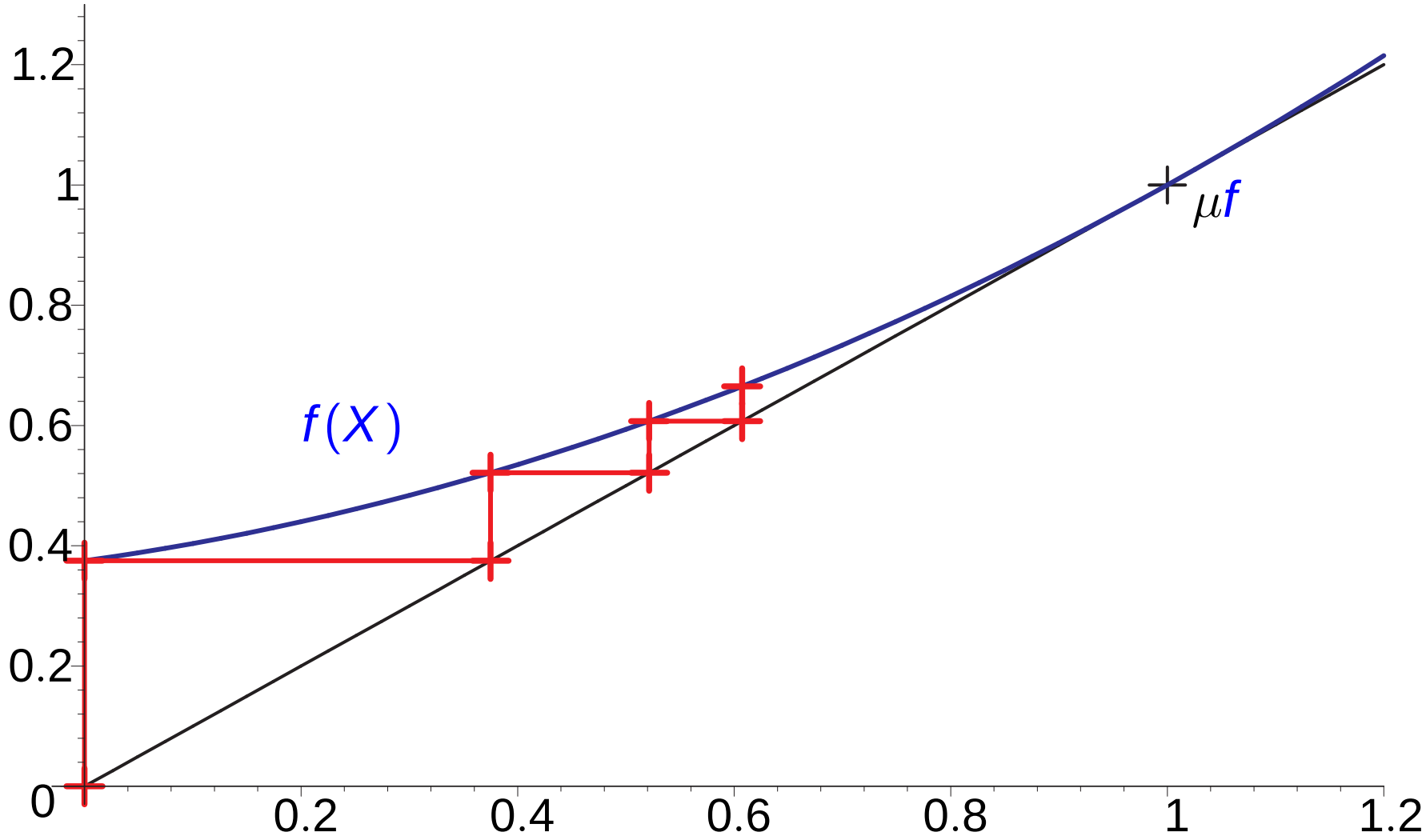
Kleene Iteration for $X = f(X)$ (univariate case)



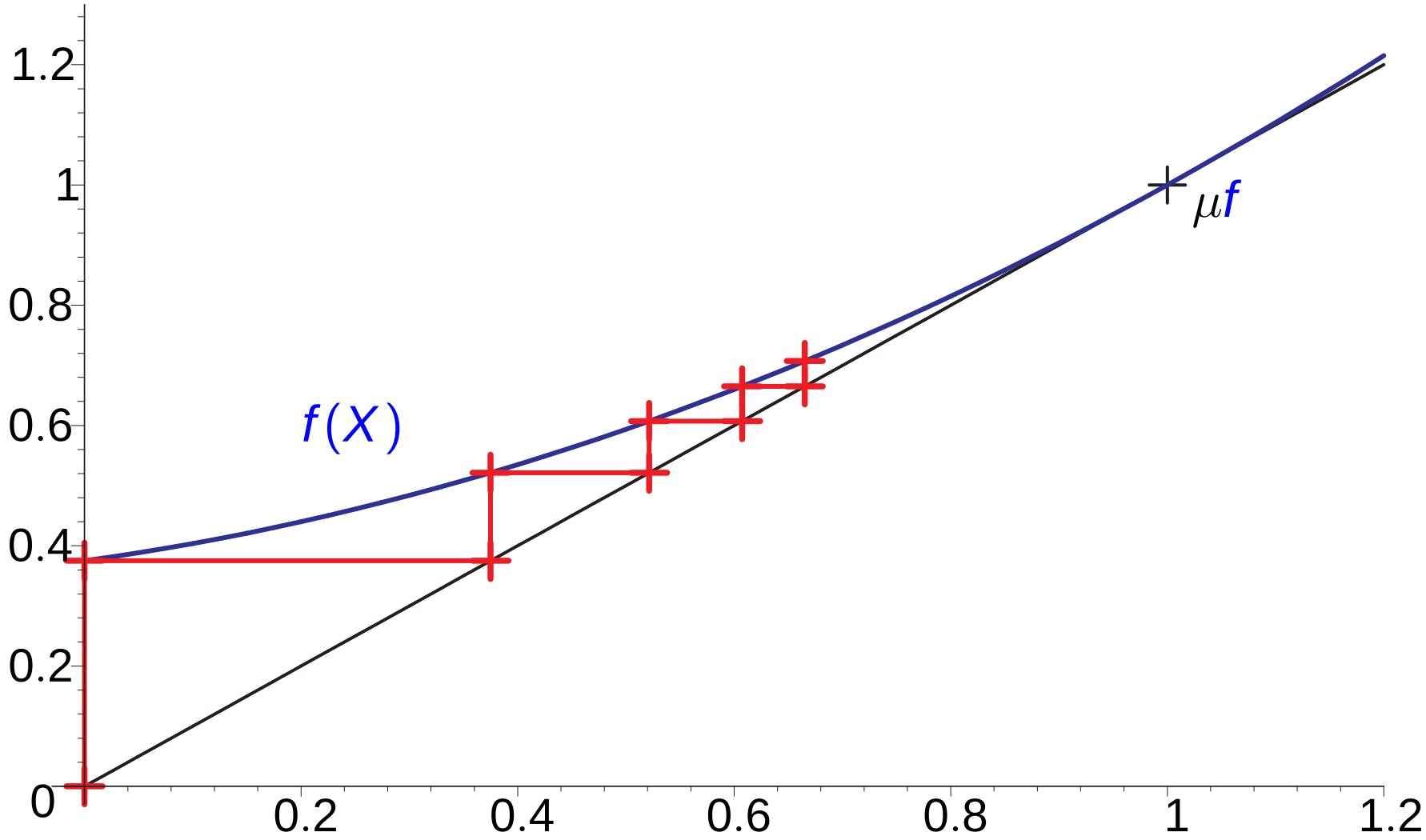
Kleene Iteration for $X = f(X)$ (univariate case)



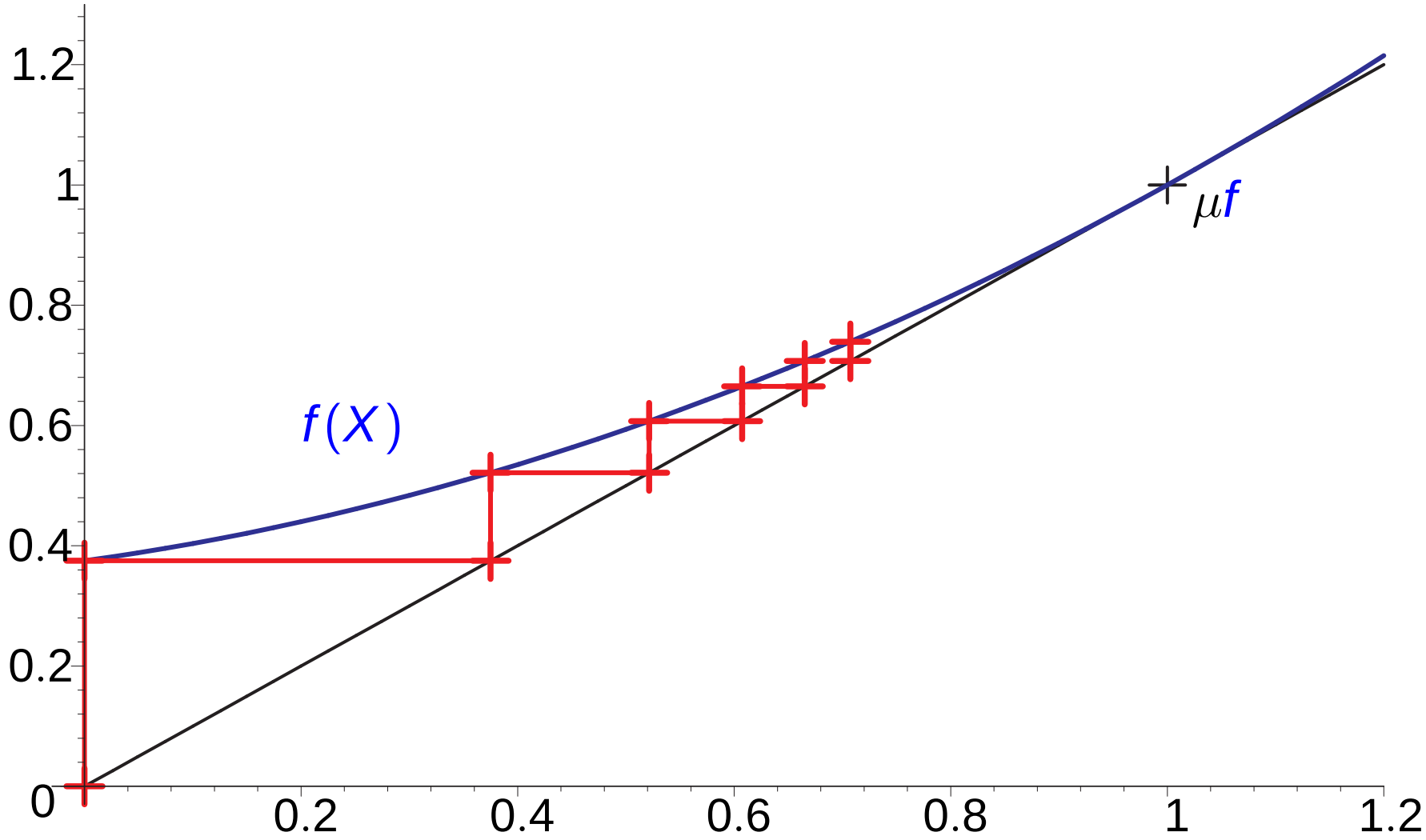
Kleene Iteration for $X = f(X)$ (univariate case)



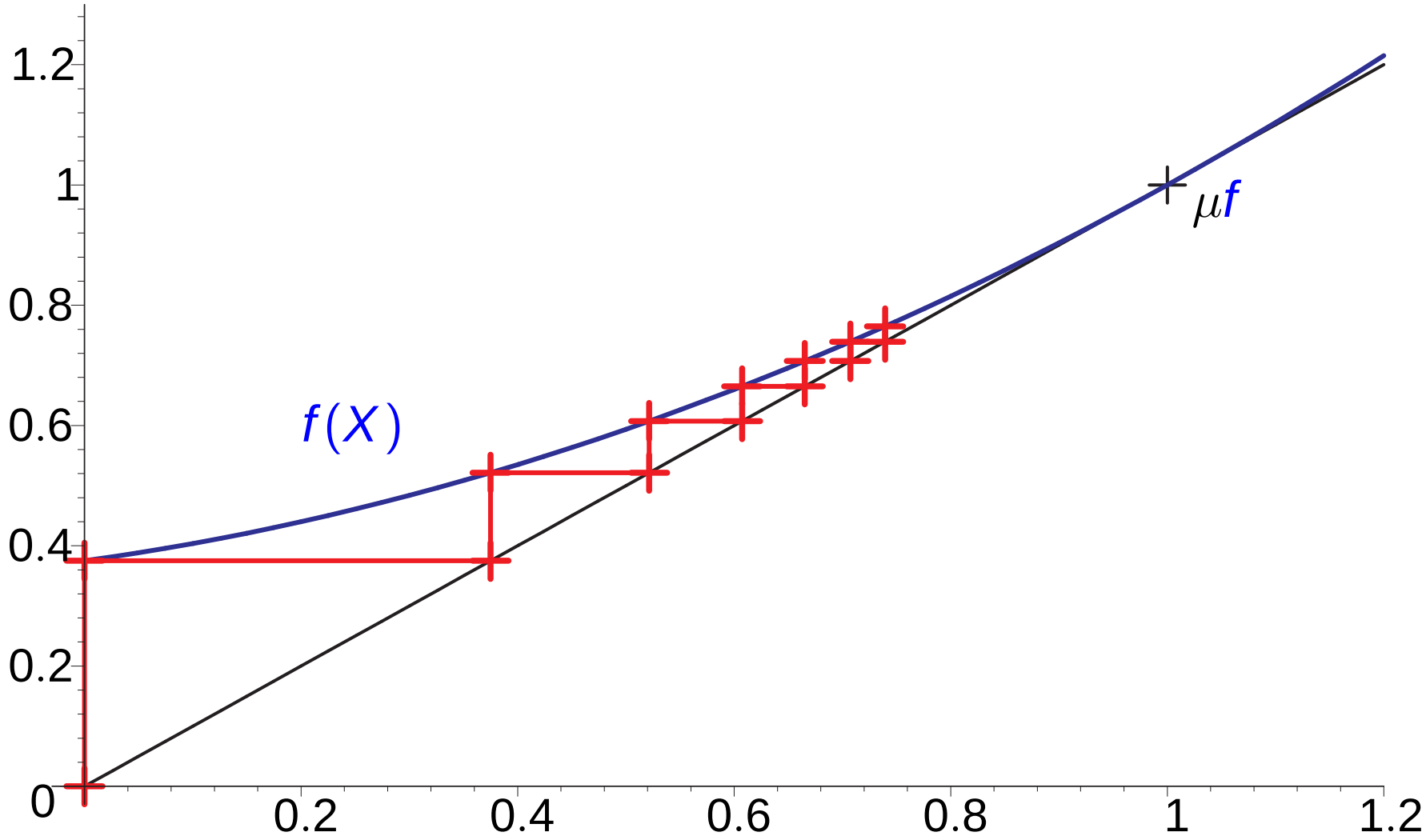
Kleene Iteration for $X = f(X)$ (univariate case)



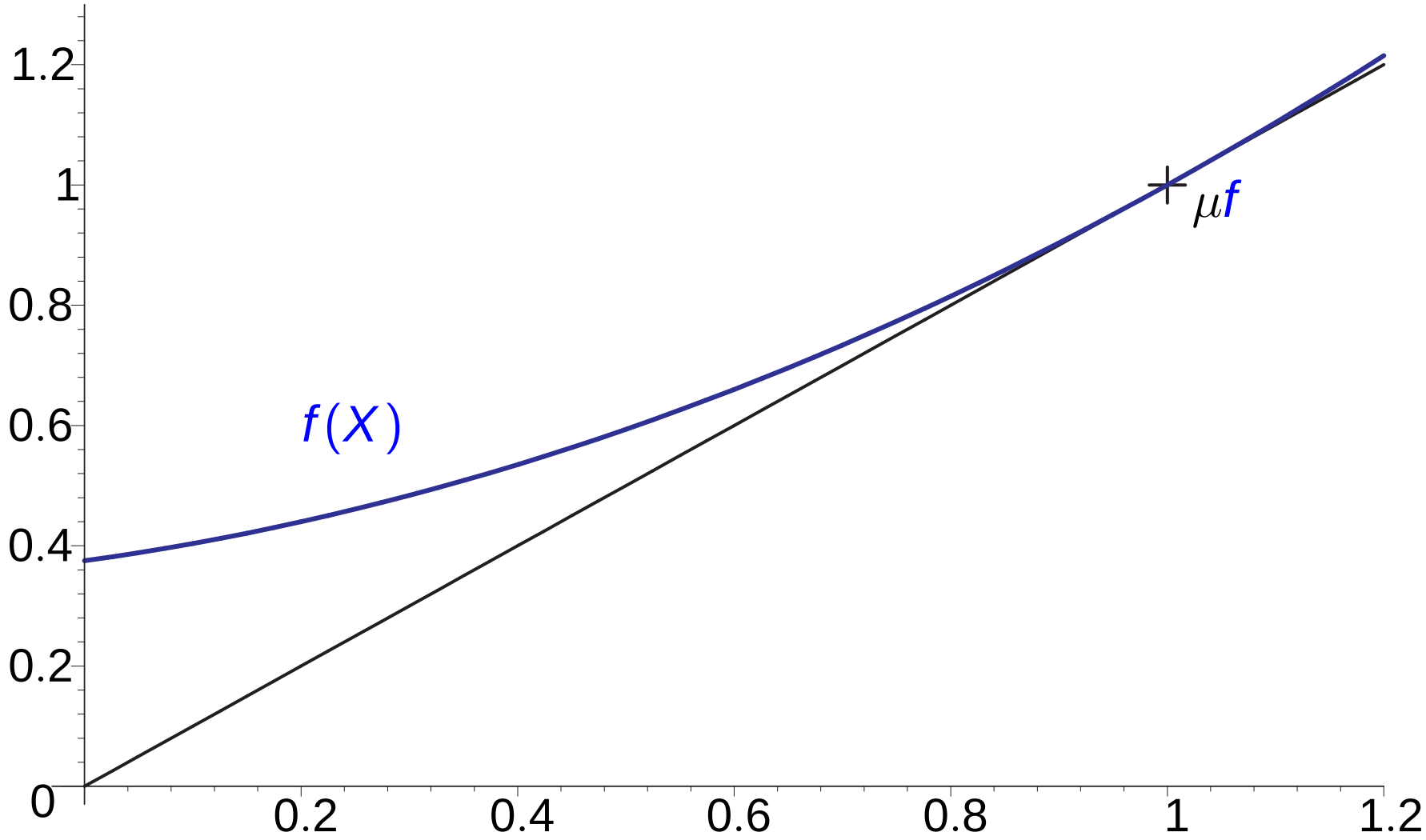
Kleene Iteration for $X = f(X)$ (univariate case)



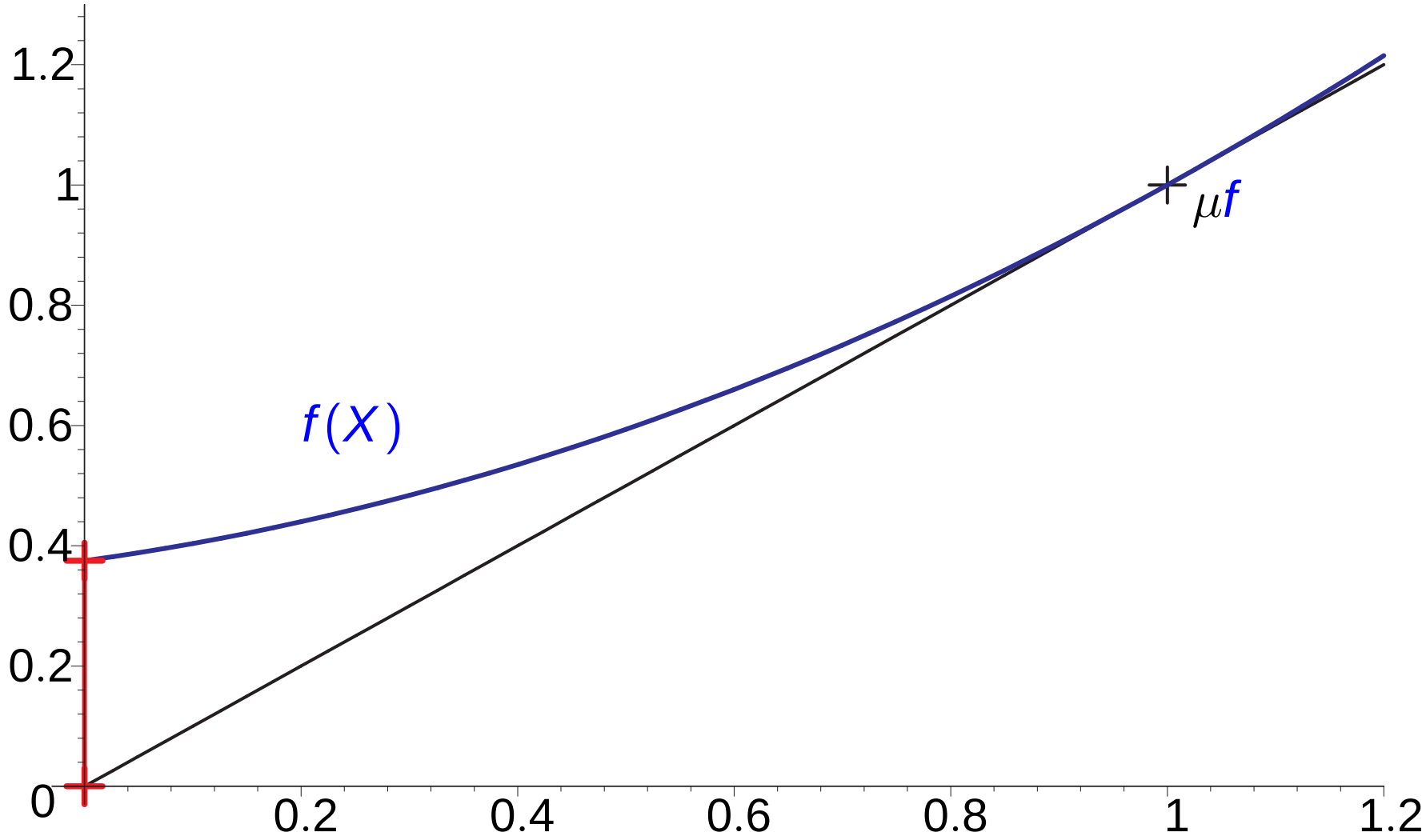
Kleene Iteration for $X = f(X)$ (univariate case)



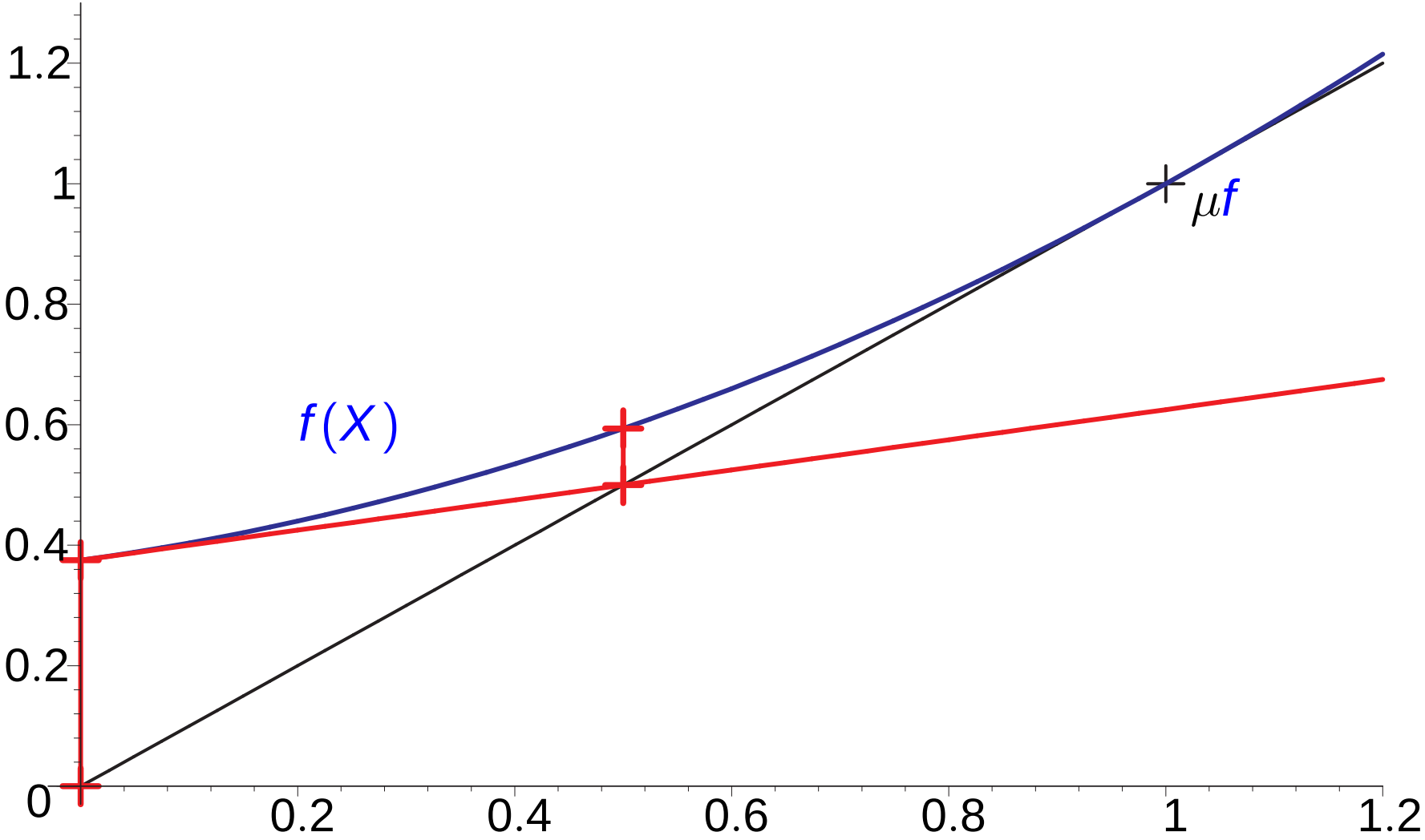
Newton's Method for $X = f(X)$ (univariate case)



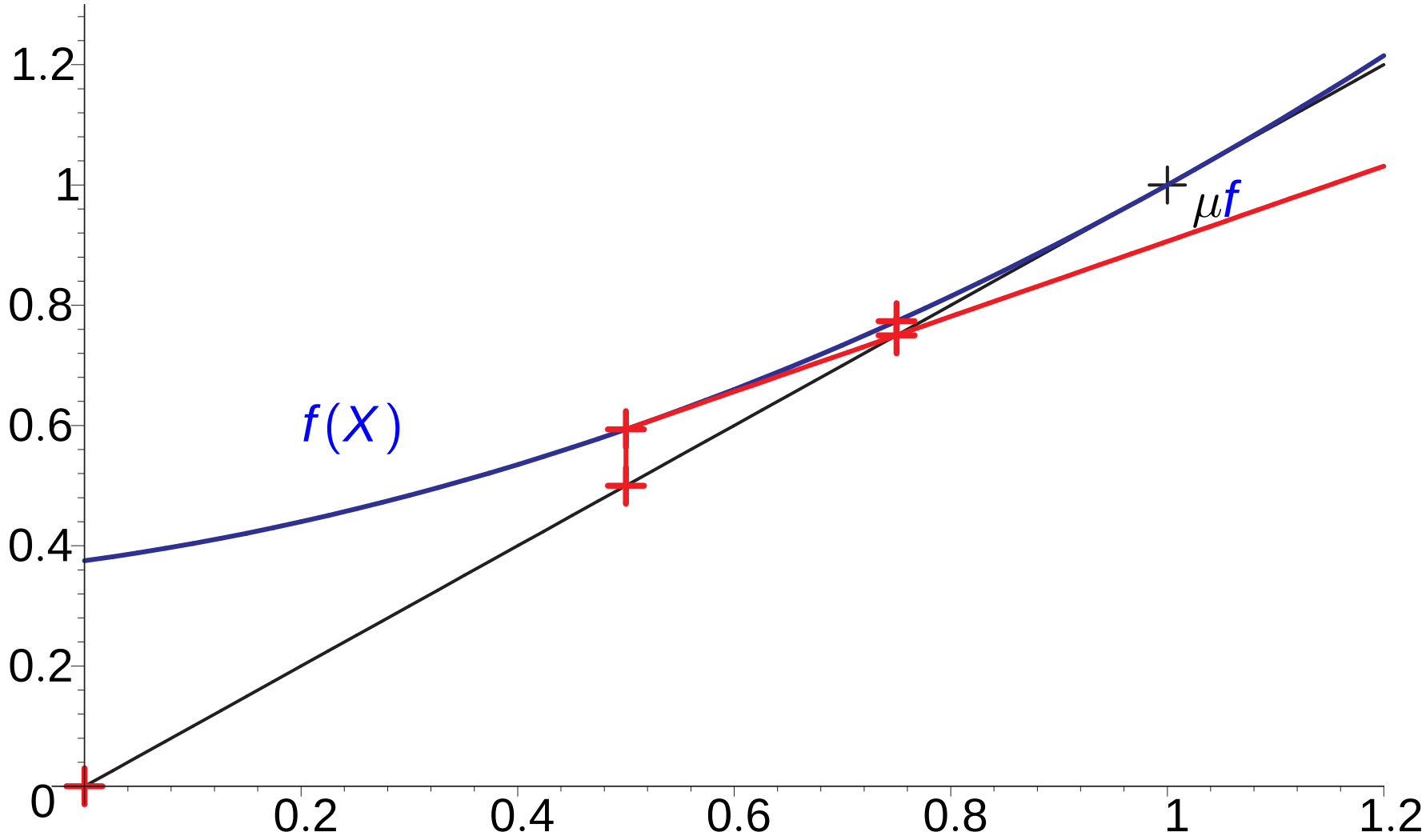
Newton's Method for $X = f(X)$ (univariate case)



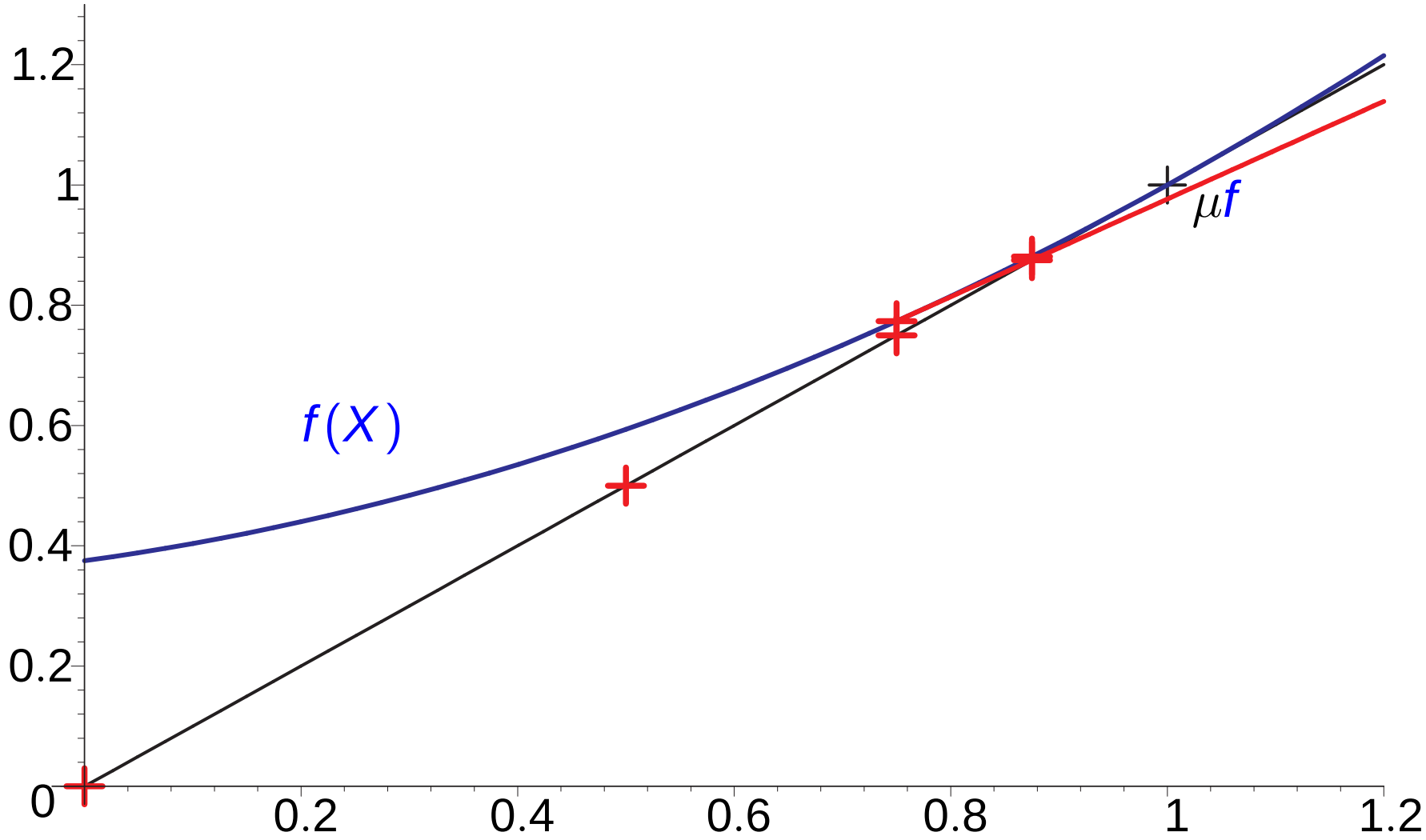
Newton's Method for $X = f(X)$ (univariate case)



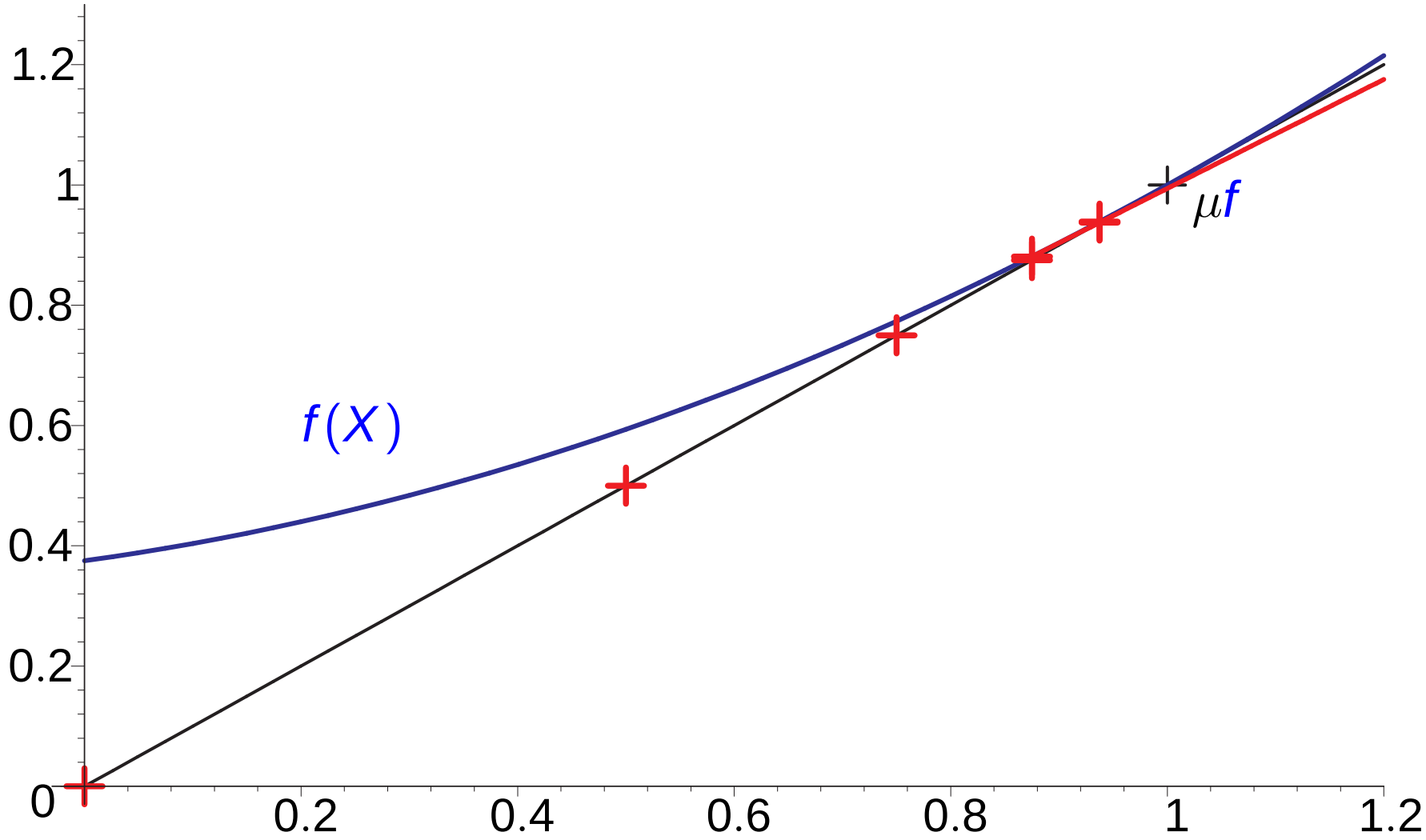
Newton's Method for $X = f(X)$ (univariate case)



Newton's Method for $X = f(X)$ (univariate case)



Newton's Method for $X = f(X)$ (univariate case)



Kleene vs. Newton

Program analysis:

- Kleene iteration is applicable to arbitrary ω-continuous semirings
- . . . but converges slowly.

Numerical mathematics:

- Newton's Method converges fast
- . . . but can only be applied to the real field

Kleene vs. Newton

Program analysis:

- Kleene iteration is applicable to arbitrary ω-continuous semirings
- . . . but converges slowly.

Numerical mathematics:

- Newton's Method converges fast
- . . . but can only be applied to the real field

Can Newton's method be generalized to arbitrary ω-continuous semirings?

Mathematical formulation of Newton's Method

Elementary analysis yields for the i-th Newton iterant ν_i:

$$\begin{aligned} \nu_0 &= 0 \\ \nu_{i+1} &= \nu_i + \Delta_i \end{aligned}$$

where Δ_i least solution of $X = Df_{\nu_i}(X) + f(\nu_i) - \nu_i$

$Df_{\nu_i}(X)$ differential of $f(X)$ at the point ν_i

Generalizing Newton's method

Key point: generalize $X = Df_\nu(X) + f(\nu) - \nu$

to arbitrary ω-continuous semirings

In an arbitrary ω-continuous semiring

- neither the differential $Df_\nu(X)$, nor
- the difference $f(\nu) - \nu$

are defined.

Overcoming the obstacles

(1) Use the algebraic definition of differential (recall that we only have polynomial functions!)

$$Df(X) = \begin{cases} 0 & \text{if } f(X) = c \\ X & \text{if } f(X) = X \\ Dg(X) + Dh(X) & \text{if } f(X) = g(X) + h(X) \\ Dg(X) \cdot h(X) + g(X) \cdot Dh(X) & \text{if } f(X) = g(X) \cdot h(X) \end{cases}$$

(2) Replace $f(\nu_i) - \nu_i$ by any δ_i such that $f(\nu_i) = \nu_i + \delta_i$

Define Δ_i as the least solution of $X = Df_{\nu_i}(X) + \delta_i$

Idempotent and commutative semirings

Theorem [Hopkins-Kozen LICS '99]: The least fixed point of a system $X = f(X)$ of n equations over an idempotent and commutative ω-continuous semiring is reached by the sequence

$$\begin{aligned} \nu_0 &= f(0) \\ \nu_{i+1} &= J(\nu_i)^* \cdot f(\nu_i) \end{aligned}$$

after at most $O(3^n)$ iterations.

Idempotent and commutative semirings

Theorem [Hopkins-Kozen LICS '99]: The least fixed point of a system $X = f(X)$ of n equations over an idempotent and commutative ω-continuous semiring is reached by the sequence

$$\begin{aligned} \nu_0 &= f(0) \\ \nu_{i+1} &= J(\nu_i)^* \cdot f(\nu_i) \end{aligned}$$

after at most $O(3^n)$ iterations.

Theorem: This is exactly Newton's sequence.

Moreover, the fixed point is reached after at most n iterations.

Two (quick) examples of application

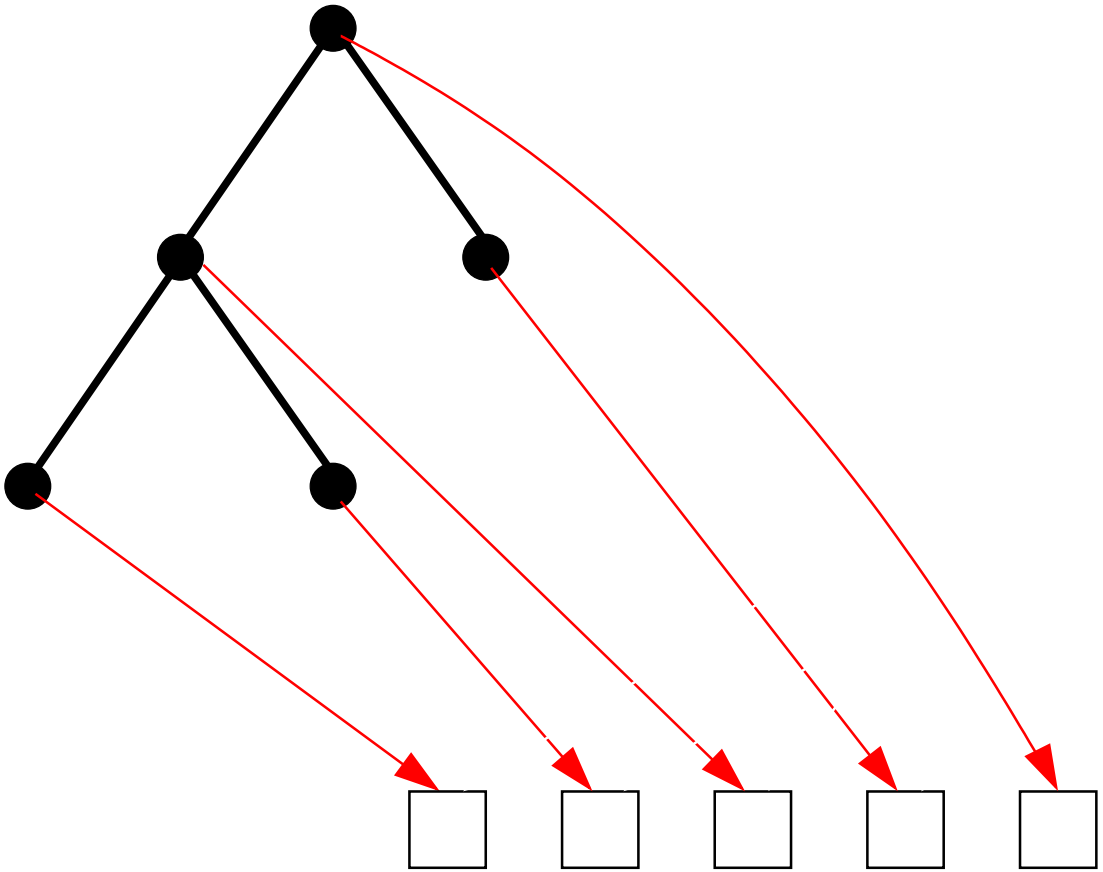
May-Alias Analysis

`listify()`: transforms a binary tree of pointers into a list of pointers by reading the tree in preorder.

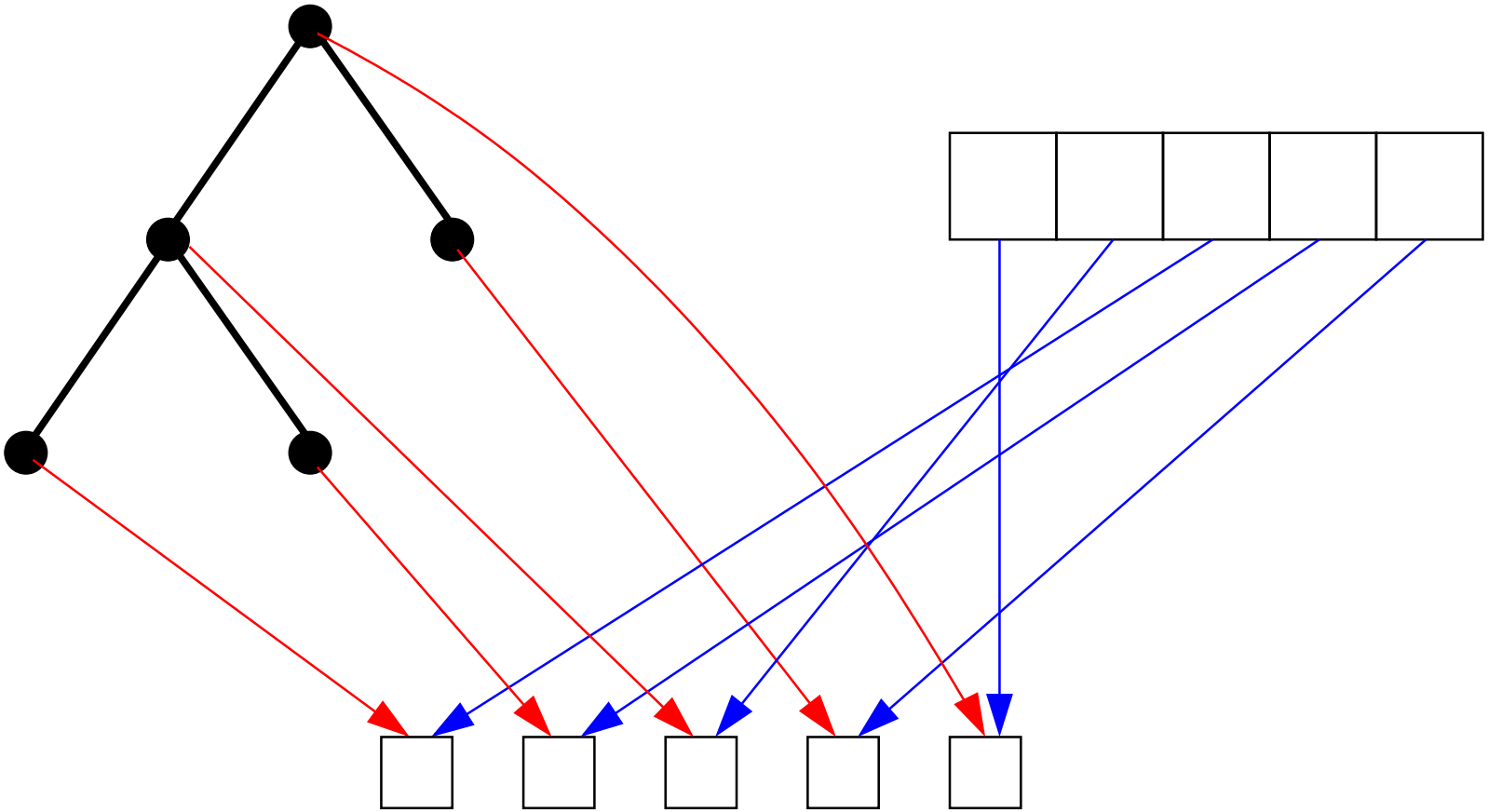
```
void listifyL() {
  T.move_left();
  listify();
  T.move_up();
}
```

```
void listify() {
  L.push_back(T→get_data());
  if( T.is_leaf() == false ) {
    listifyL(); listifyR();
  }
}
```

May-Alias Analysis



May-Alias Analysis



May-Alias Analysis

Which data access paths of tree and list may point to the same element ?

Access path of the tree: word $w_1 \in \{\texttt{left}, \texttt{right}\}^*$

Access path of the list: word $w_2 \in \{\texttt{next}\}^*$

May-alias information: set of pairs (w_1, w_2) such that w_1, w_2 may point to the same cell

Commutative abstraction: $(\texttt{left right left}, \texttt{next next}) \rightarrow (3, 1, 2)$

May-Alias Analysis

Kleene iteration does not terminate.

Newton's method terminates after one iteration and provides the following information:

- Data access paths with 0 `right`'s and ℓ `left`'s may only be aliased to the ℓ-th element of the list.
- Data access paths with r `right`'s and ℓ `left`'s may only be aliased to the (2r + ℓ)-th element of the list, or to larger elements of the same parity.

Lazy evaluation of And-Or trees

Nodes are only constructed and evaluated (to 0 or 1) if needed.
(e.g., if left subtree of And-node evaluates to 1, right subtree is not constructed)

```
function And(node)
 if node.leaf() then
  return node.value()
 else
  v := Or(node.left)
  if v = 0 then
   return 0
  else
   return Or(node.right)
```

```
function Or(node)
 if node.leaf() then
  return node.value()
 else
  v := And(node.left)
  if v = 1 then
   return 1
  else
   return And(node.right)
```

Assume the probabilities that node.leaf() returns **true** and node.value() returns **1** are both $1/2$.

We perform an analysis to compute the average runtime.

Kleene vs. Newton

Neither Kleene nor Newton terminate, but Newton converges faster:

i	$k^{(i)}$ And_0	$\nu^{(i)}$ And_0	$k^{(i)}$ And_1	$\nu^{(i)}$ And_1
0	2.000	2.000	2.000	2.000
1	2.538	3.588	2.333	3.383
2	2.913	5.784	3.012	5.906
3	3.429	6.975	3.381	7.194
4	3.793	7.067	3.904	7.295

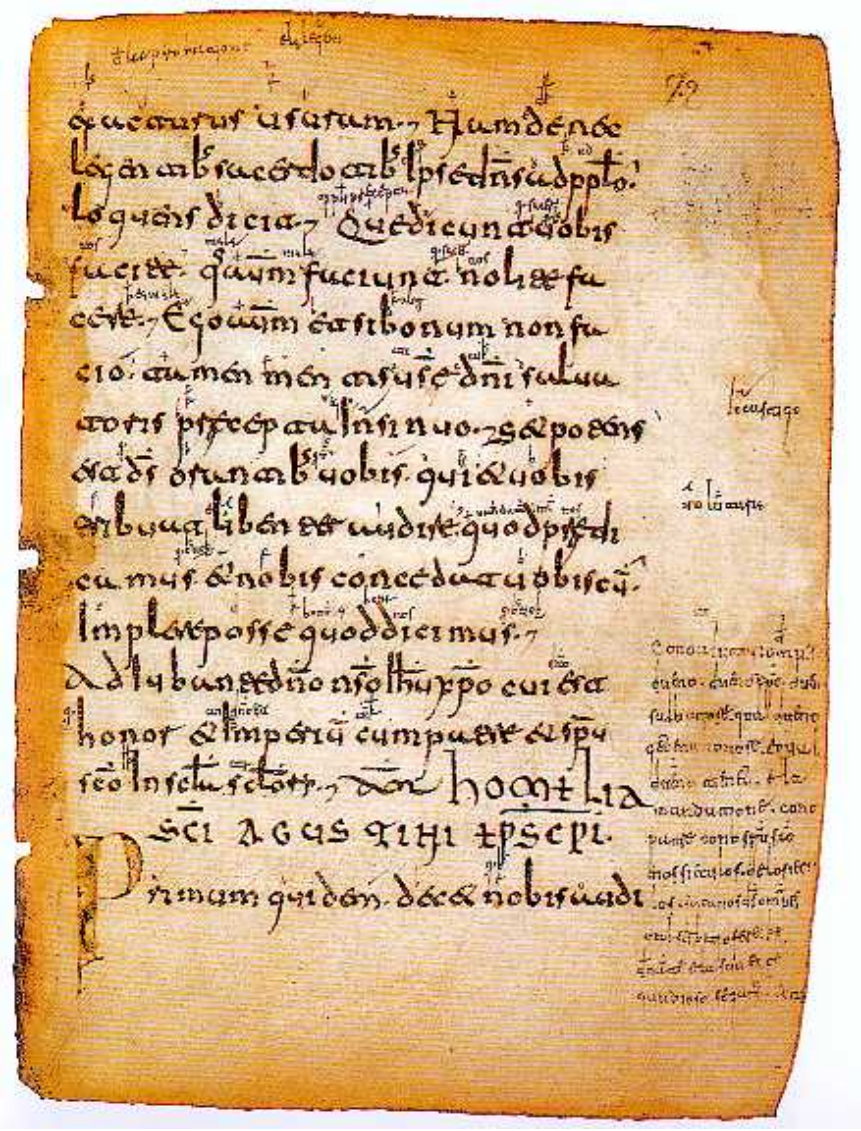
Messages of this talk

The theory of solving “program analysis equations” is not be as well understood as we thought.

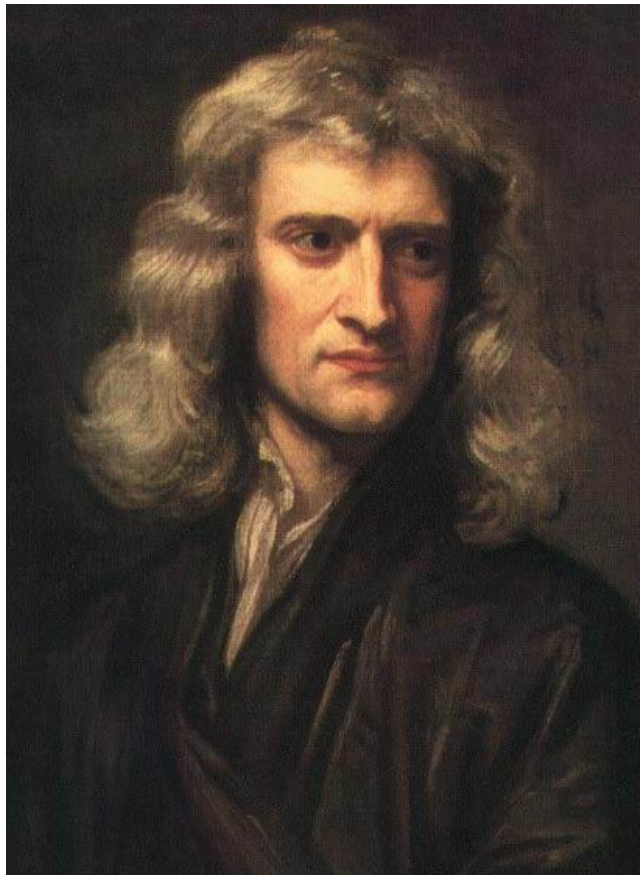
We can learn from numerical mathematics and contribute to it.

Go for unified theory of qualitative and quantitative program analysis.

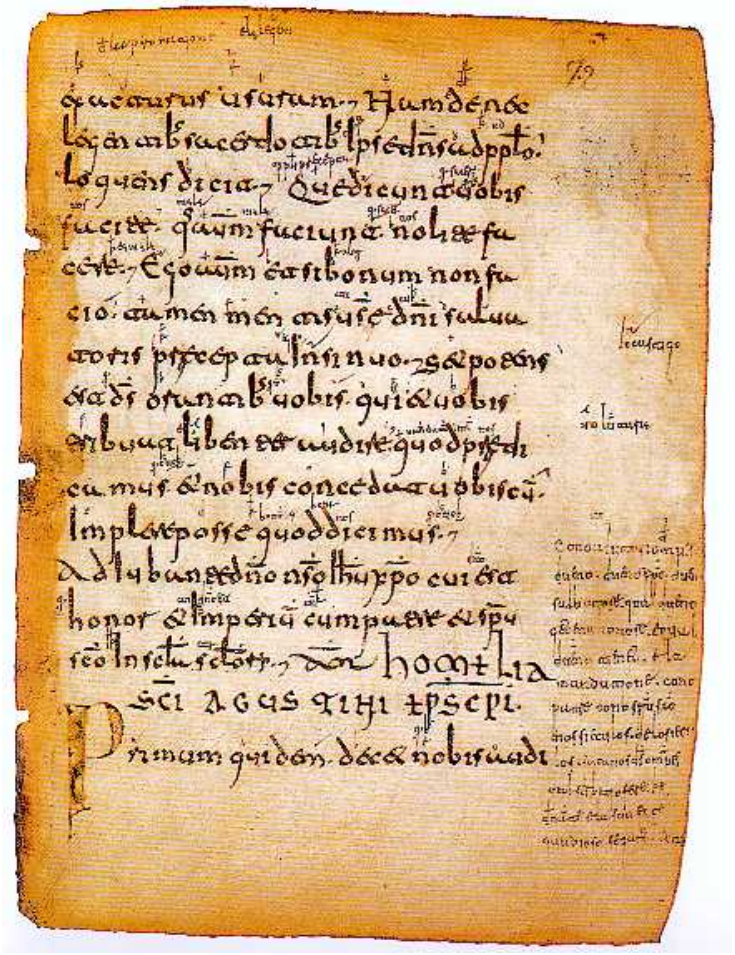
Commenting the works …



. . . of the giants



And yet . . .



The glosses of Saint Emilianus,
written around 1000 AC.
Oldest writing in (a form of) Spanish,
one of the most spoken languages
in the world.

JOURNAL OF COMPUTER AND SYSTEM SCIENCES 5, 511–523 (1971)

Marked Directed Graphs*

F. COMMONER AND A.W. HOLT

Applied Data Research, Wakefield, Massachusetts 01880

AND

S. EVEN, A. PNUELI

The Weizmann Institute of Science, Rehovot, Israel

Received October 6, 1970

Shimon Even, 1935–2004

Amir Pnueli, 1941–2009

I. INTRODUCTION

Diverse graph structure models for concurrent processing systems have been suggested and used. The structures differ in generality and scope according to the properties one wishes to model and analyze. In this paper we solve a problem of maximal storage requirements for a simple flowchart model called the Marked Graph Model.