Water computing: A P system variant

Alec Henderson[1], Radu Nicolescu[1], Michael J. Dinneen[1], TN Chan[2], Hendrik Happe[3], and Thomas Hinze[3]

[1]School of Computer Science
The University of Auckland, Auckland, New Zealand

[2]Compucon New Zealand, Auckland, New Zealand

[3]Department of Bioinformatics
Friedrich Schiller University of Jena, Jena, Germany

September 10, 2020

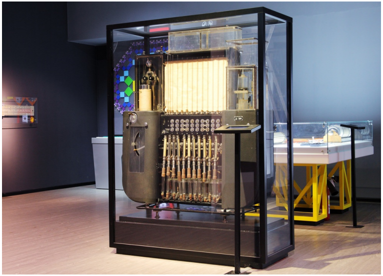
Outline of presentation

The power of water

Coromandel, New Zealand

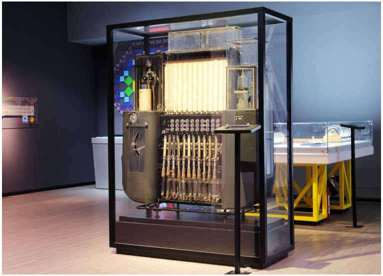
The magical sound,
of the cascading water,
natural beauty

Water Integrator



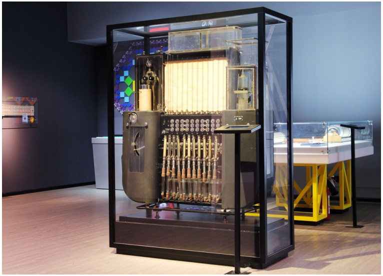
- First model built in 1936, in USSR; modular model in 1941, standard unified units in 1949-1955

Water Integrator



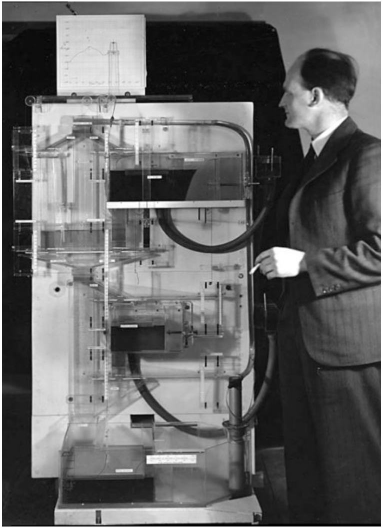
- First model built in 1936, in USSR; modular model in 1941, standard unified units in 1949-1955
- Used to solve inhomogeneous differential equations with applications such as: solving construction issues in the sands of Central Asia and in permafrost and in studying the temperature regime of the Antarctic ice sheet

Water Integrator



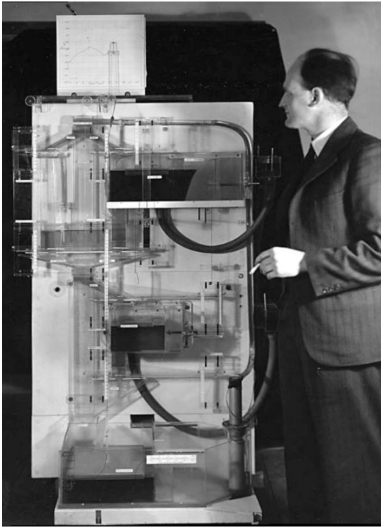
- First model built in 1936, in USSR; modular model in 1941, standard unified units in 1949-1955
- Used to solve inhomogeneous differential equations with applications such as: solving construction issues in the sands of Central Asia and in permafrost and in studying the temperature regime of the Antarctic ice sheet
- Only surpassed by digital computers in the 1980's.

MONIAC (Monetary National Income Analogue Computer)



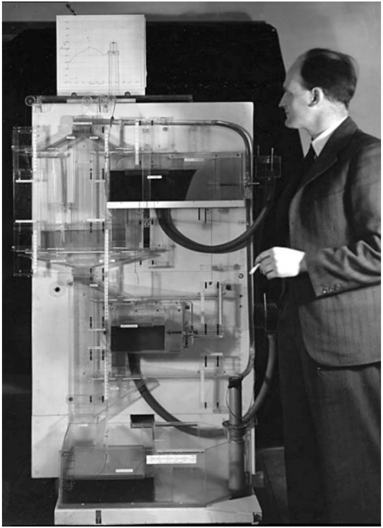
- MONIAC (Monetary National Income Analogue Computer) also known as the Phillips Hydraulic Computer and the Financephalograph

MONIAC (Monetary National Income Analogue Computer)



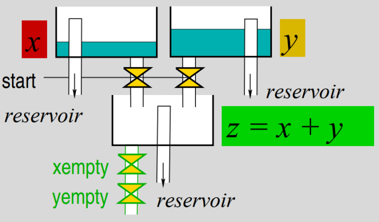
- MONIAC (Monetary National Income Analogue Computer) also known as the Phillips Hydraulic Computer and the Financephalograph
- First built in 1949 by New Zealand economist Bill Phillips to model the UK economy.

MONIAC (Monetary National Income Analogue Computer)



- MONIAC (Monetary National Income Analogue Computer) also known as the Phillips Hydraulic Computer and the Financephalograph
- First built in 1949 by New Zealand economist Bill Phillips to model the UK economy.
- Built as a teaching aid it was discovered that it was also an effective economic simulator.

Previous Work[1]



- No centre of control.
- Water flows if and only if all valves on a pipe are open.
- Water flows between tanks concurrently.

[1] Thomas Hinze et al. "Membrane computing with water". In: *J. Membr. Comput.* 2.2 (2020), pp. 121–136.

Open Problems

- How can functions be stacked without a combinatorial explosion of the number of valves?

Open Problems

- How can functions be stacked without a combinatorial explosion of the number of valves?
- How can termination of the system be detected?

Open Problems

- How can functions be stacked without a combinatorial explosion of the number of valves?
- How can termination of the system be detected?
- How to reset the system?

Open Problems

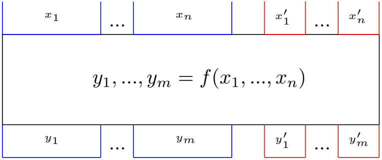
- How can functions be stacked without a combinatorial explosion of the number of valves?
- How can termination of the system be detected?
- How to reset the system?
- Is the system Turing complete?

Open Problems

- How can functions be stacked without a combinatorial explosion of the number of valves?
- How can termination of the system be detected?
- How to reset the system?
- Is the system Turing complete?

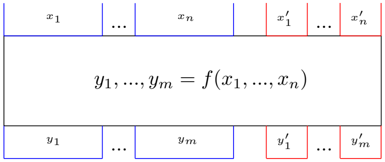
We solve these problems by introducing a set of control tanks.

Control tanks



- A control tank for each input and output.

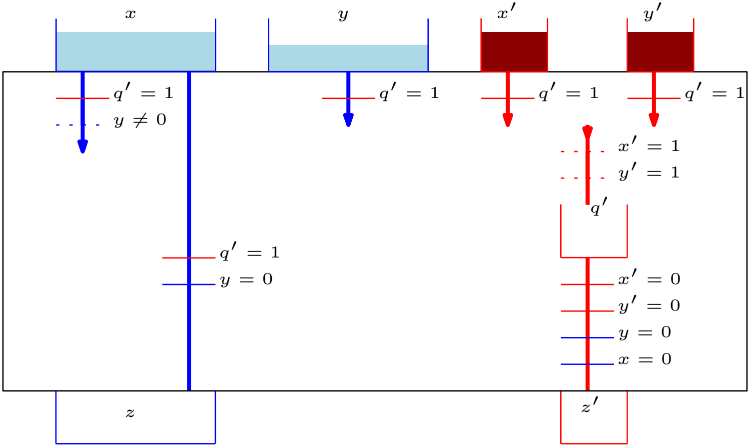
Control tanks



- A control tank for each input and output.
- Start the computation once all control tanks are filled.
- An output is ready when it's control tank is full.

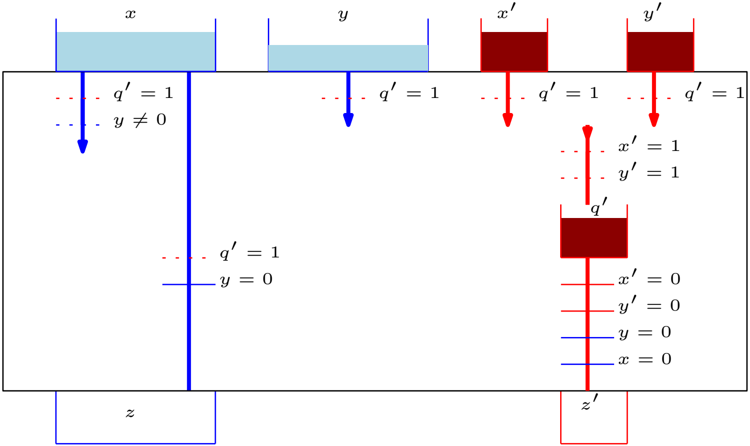
Subtraction

$z = x - y$



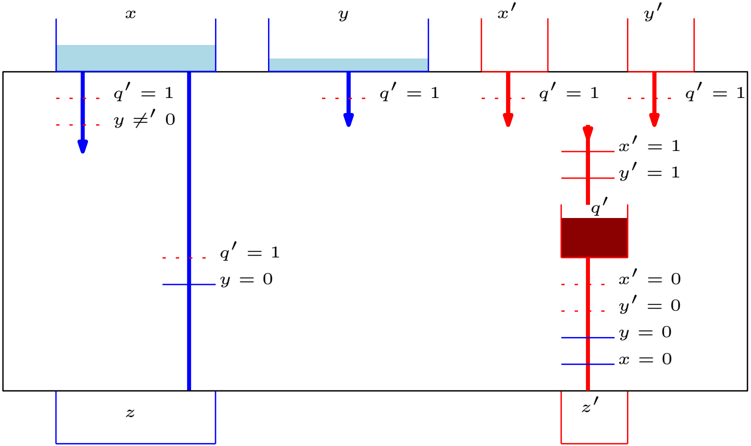
Subtraction

$$z = x - y$$



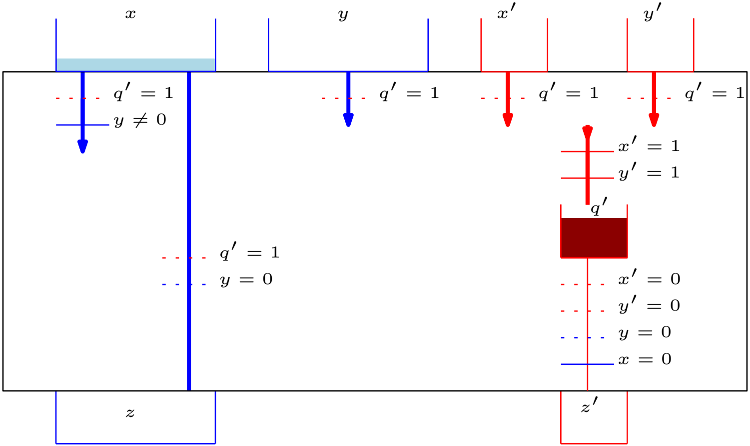
Subtraction

$z = x - y$



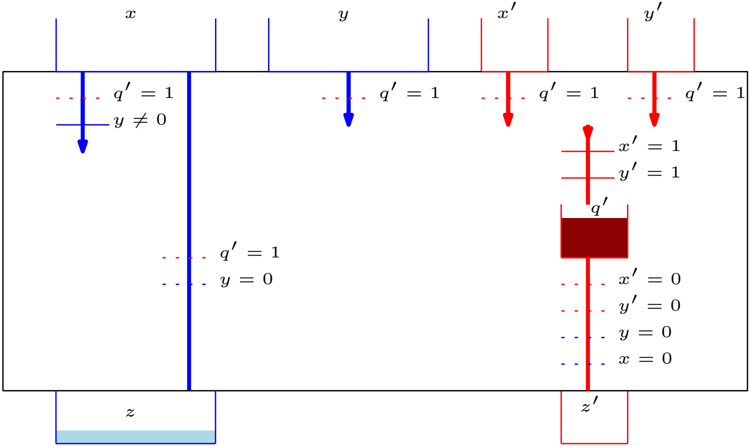
Subtraction

$$z = x - y$$



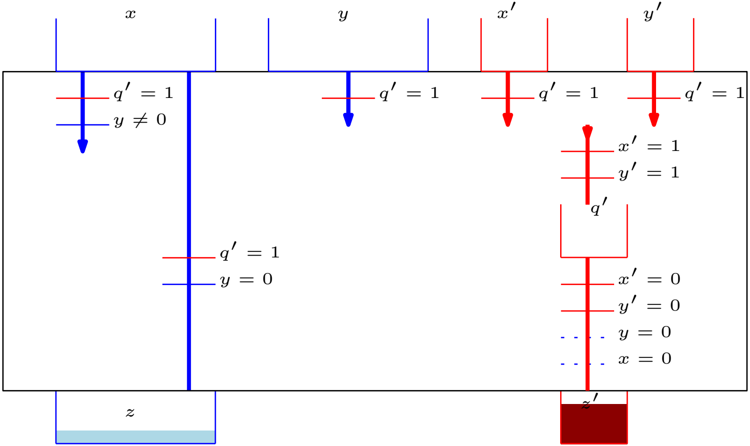
Subtraction

$$z = x - y$$



Subtraction

$$z = x - y$$



Primitive Recursion

To prove our system can construct all unary primitive recursive functions we use the following base functions and closure operators[2]:

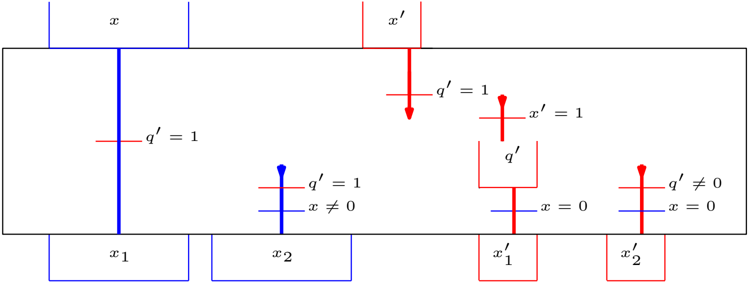
- **Successor function**: $S(x) = x + 1$
- **Subtraction function**: $B(x, y) = x - y$
- **Composition operator**: $C(h, g)(x) = h(g(x))$
- **Difference operator**: $D(f, g)(x) = f(x) - g(x)$
- **Primitive recursion operator**: $P(f)(0) = 0$, $P(x + 1) = f(P(x))$

To assist in the proof we also construct two copy functions: inplace $i(x)$ and destructive $c(x)$.

[2] Cristian Calude and Lila Sântean. "On a theorem of Günter Asser". In: *Mathematical Logic Quarterly* 36.2 (1990), pp. 143–147.

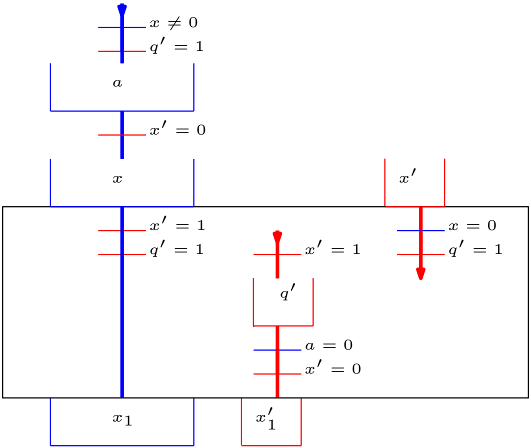
Destructive copy $x_1 = x_2 = x$

$x_1 = x_2 = x$



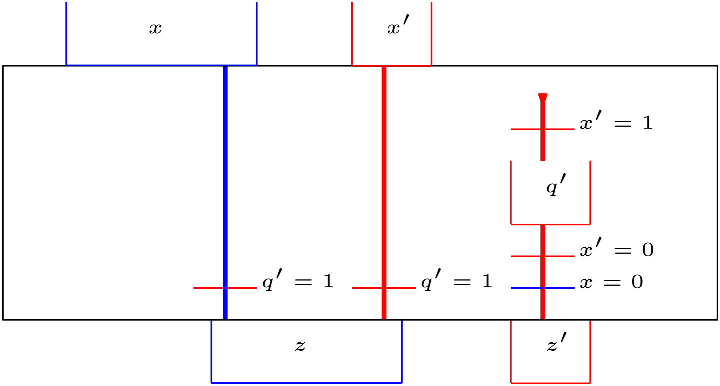
Inplace copy $x_1 = x$

$x_1 = x$

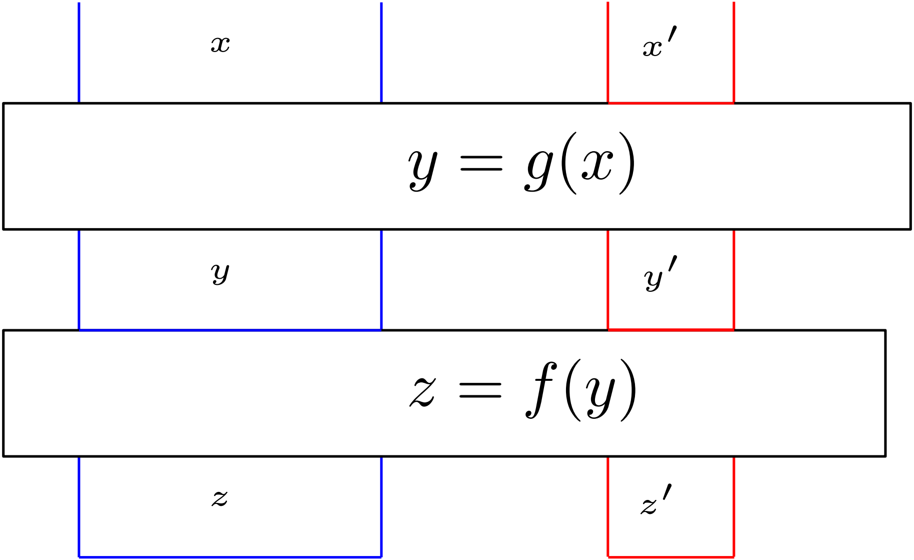


Successor function $S(x) = x + 1$

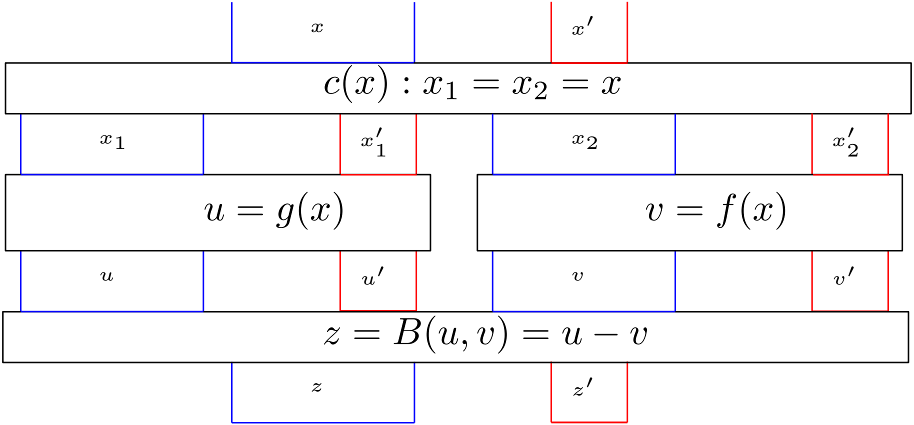
$$z = x + 1$$



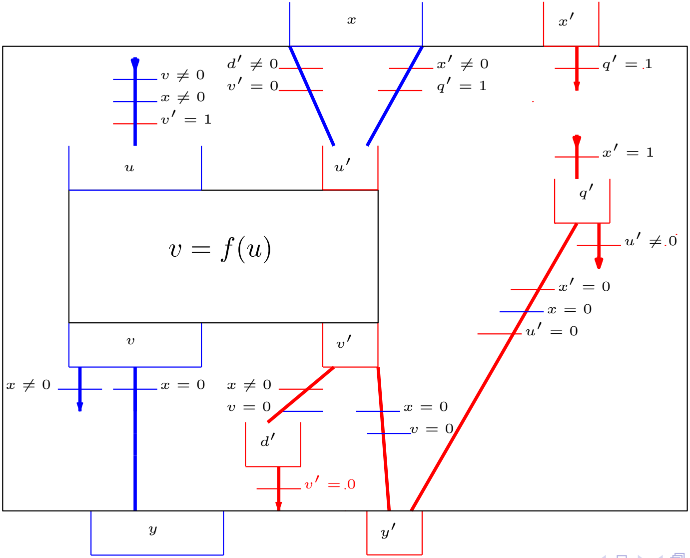
Composition operator $C(h, g)(x) = h(g(x))$



Difference operator $D(f,g)(x) = f(x) - g(x)$



Primitive recursion operator $P(f)(0) = 0$, $P(x+1) = f(P(x))$

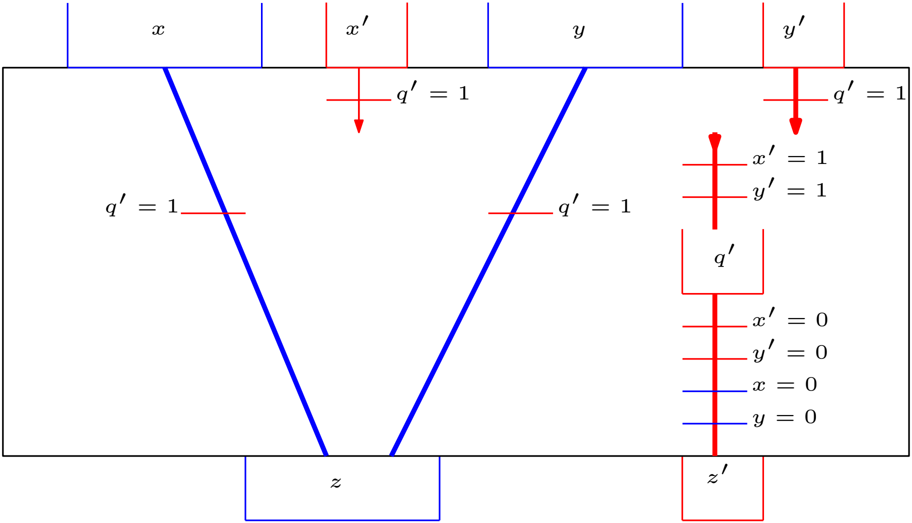


To prove Turing completeness we require that our system can construct the unary primitive recursive functions as well as[3]:

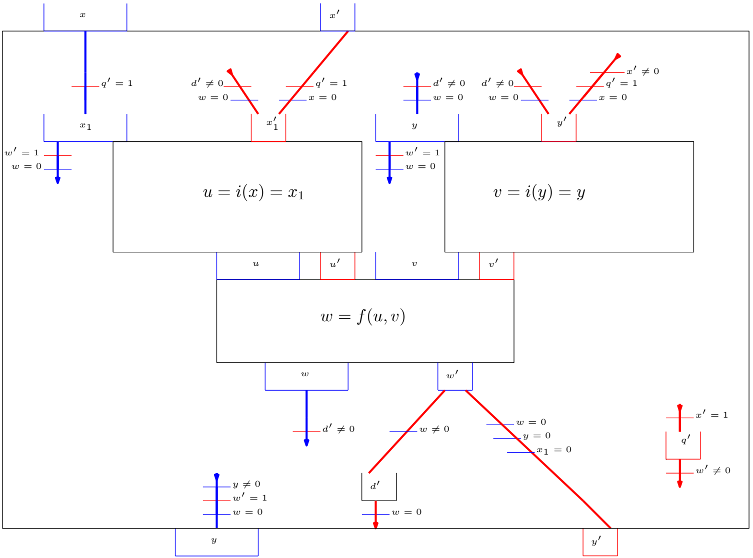
- **Addition function**: $A(x, y) = x + y$
- μ **operator**: $\mu_y(f)(x, y) = \min_y\{f(x, y) = 0\}$

[3] Julia Robinson. "General recursive functions". In: *Proceedings of the american mathematical society* 1.6 (1950), pp. 703–718.

Addition function $A(x, y) = x + y$



μ operator $\mu_y(f)(x, y) = \min_y\{f(x, y) = 0\}$



A restricted cP system

- cP systems subcells act as data storage the same as our water tanks.

A restricted cP system

- cP systems subcells act as data storage the same as our water tanks.
- cP systems have a set of rules for changing the content in the subcells. Similarly we have a set of valves and pipes.

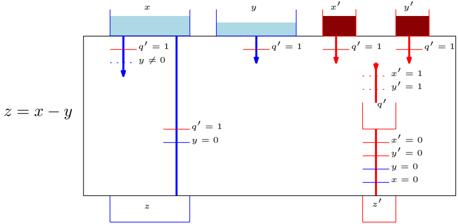
A restricted cP system

- cP systems subcells act as data storage the same as our water tanks.
- cP systems have a set of rules for changing the content in the subcells. Similarly we have a set of valves and pipes.
- cP systems are able to create and consume subcells whereas in our system we cannot create and consume tanks.

A restricted cP system

- cP systems subcells act as data storage the same as our water tanks.
- cP systems have a set of rules for changing the content in the subcells. Similarly we have a set of valves and pipes.
- cP systems are able to create and consume subcells whereas in our system we cannot create and consume tanks.
- Using the similarities we are able to construct cP system-like rules which don't contain creating or consuming of subcells.

Rules for subtraction



$s_1\ q()$	$\rightarrow$	$s_2\ q(1)$	$\mid\ c_x(1)\ c_y(1)$	(1)
$s_2\ c_x(1)$	$\rightarrow$	$s_2\ c_x()$	$\mid\ q(1)$	(2)
$s_2\ c_y(1)$	$\rightarrow$	$s_2\ c_y()$	$\mid\ q(1)$	(3)
$s_2\ v_x(X1)$	$\rightarrow$	$s_2\ v_x(X)$	$\mid\ q(1)\ v_y(\_1)$	(4)
$s_2\ v_y(Y1)$	$\rightarrow$	$s_2\ v_y(Y)$	$\mid\ q(1)$	(5)
$s_2\ v_x(X1)\ v_z(Z)$	$\rightarrow$	$s_2\ v_x(X)\ v_z(Z1)$	$\mid\ q(1)\ v_y()$	(6)
$s_2\ q(1)\ c_z()$	$\rightarrow$	$s_3\ q()\ c_z(1)$	$\mid\ c_x()\ c_y()$	(7)

Conclusion

- We have proven that our water tank system is Turing complete, via construction of μ-recursive functions.

Conclusion

- We have proven that our water tank system is Turing complete, via construction of μ-recursive functions.
- We have demonstrated how termination can be detected, as well as how to combine different functions without an exponential explosion of the number of valves.

Conclusion

- We have proven that our water tank system is Turing complete, via construction of μ-recursive functions.
- We have demonstrated how termination can be detected, as well as how to combine different functions without an exponential explosion of the number of valves.
- We have given a brief description on how our water tank system is a restricted version of cP systems.

Future Work

- Being able to run a function without all of the controls being filled.

Future Work

- Being able to run a function without all of the controls being filled.
- Solving practical problems with the system.

Future Work

- Being able to run a function without all of the controls being filled.
- Solving practical problems with the system.
- Using the system to model biological systems.

Future Work

- Being able to run a function without all of the controls being filled.
- Solving practical problems with the system.
- Using the system to model biological systems.
- Constructing a programmable universal water computer.

Questions

Coromandel New Zealand

- Thank you for listening.

Questions

Coromandel New Zealand

- Thank you for listening.
- Any questions?