Computational Complexity in Time Loops

Presentation by Marien Raat

Based on *Closed Timelike Curves Make Quantum and Classical Computing Equivalent* by Scott Aaronson and John Watrous (2009)

November 14, 2022

Utrecht University

Outline

Time travel

Doctor Who

Tenet (2020)

Back to the Future (1989)

What are time loops?

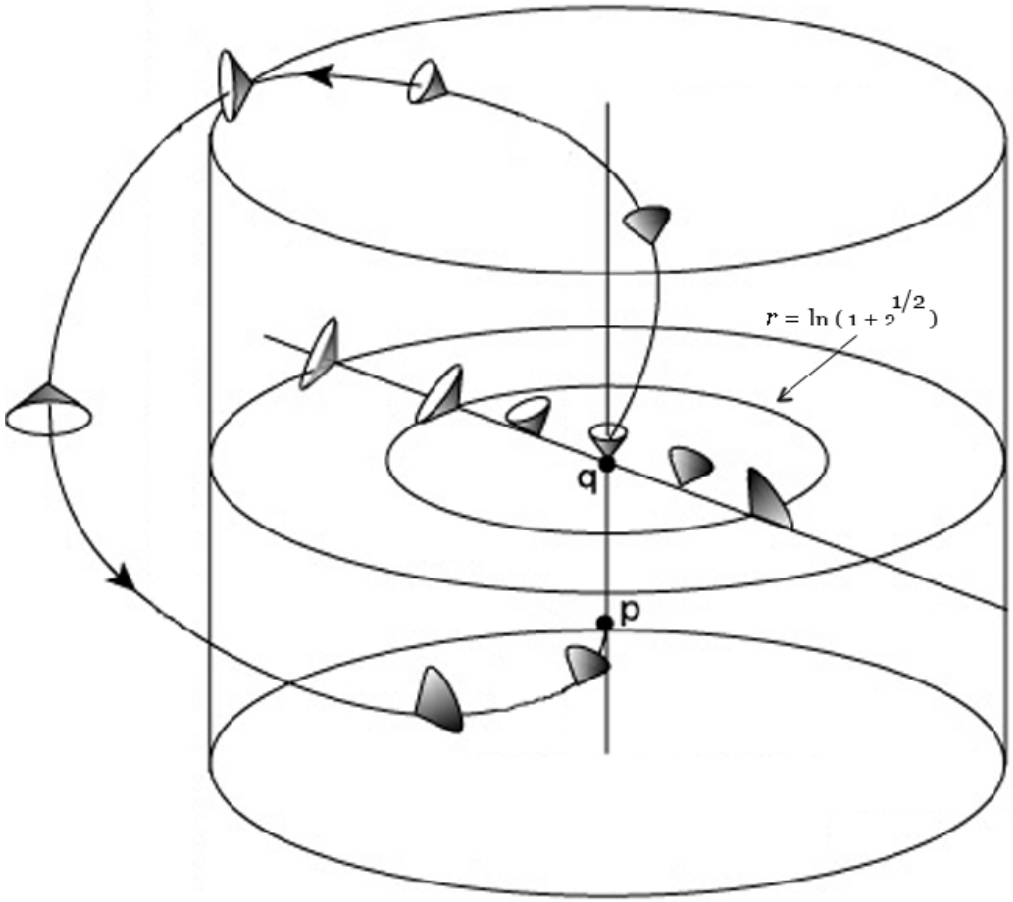

Figure 1: Space-time Illustration of a closed timelike curve

Are time loops possible?

- Einstein's General Relativity
- Rotating black holes
- Quantum Gravity
- Paradoxes?

Figure 2: Kurt Gödel in 1925

Weird $\neq$ impossible

Figure 3: First picture of a black hole

What about computation?

- Church-Turing thesis
- Physicalisation
- Consistency condition
- Reasonably sized time loop
- Reasonable = polynomial

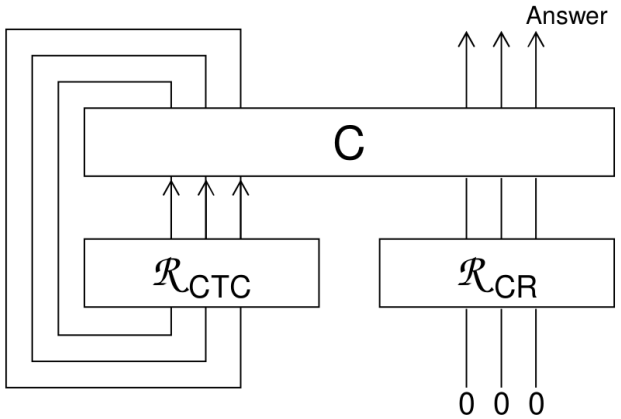

Figure 4: Illustration of Turing Machine in time loop from Aaronson and Watrous (2009)

Why talk about this?

- Interesting
- Learning about our models of computation
- What is fundamentally computable?
- Challenge assumptions about computing

Enter paradoxes

Figure 5: Marty's family disappears from the photo as he changes his past

Grandfather paradox

- Things happen once
- So they need to be consistent
- What if I travel back in time to kill my grandfather?

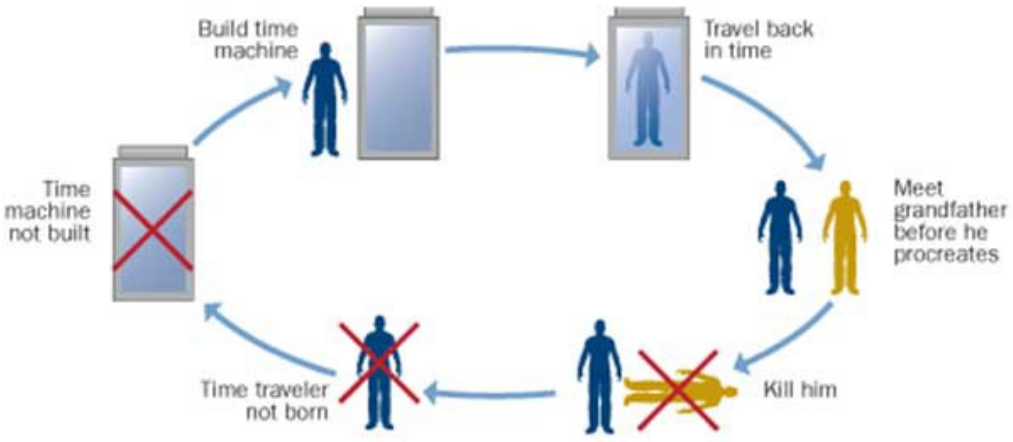

Figure 6: The grandfather paradox illustrated

Polchinski's paradox

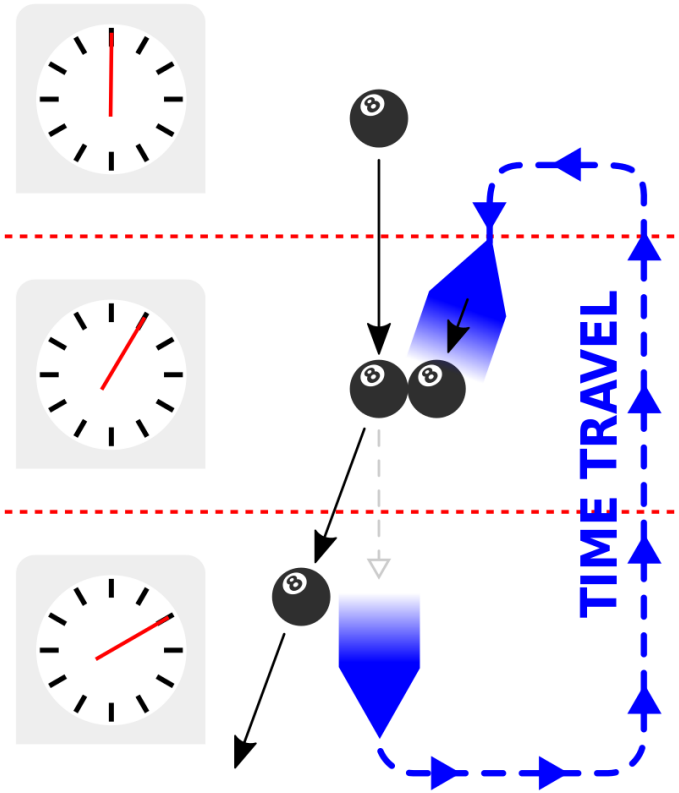

Figure 7: Polchinski's paradox

Deutsch's solution

- Quantum mechanics near closed timelike lines (1991)
- Enter quantum mechanics
- Quantum state is probabilistic
- Probability distribution needs to be consistent
- There is always a solution
- Grandfather paradox: you kill your grandfather with 50% chance

Causal consistency

Definition

A time loop is causally consistent iff the quantum state of the hypersurface at the start of the time loop is equal to the quantum state at the end of the time loop.

Theorem

Every time loop will be causally consistent

Many worlds

- Every part of the probability distribution is a world
- When we measure we move into one of the possible worlds
- Explains why we don't see probability distributions
- Popular interpretation of quantum mechanics
- *The fabric of reality* by David Deutsch
- *Many-Worlds Interpretation of Quantum Mechanics* on Stanford Encyclopedia of Philosophy on

Grandfather paradox in many worlds

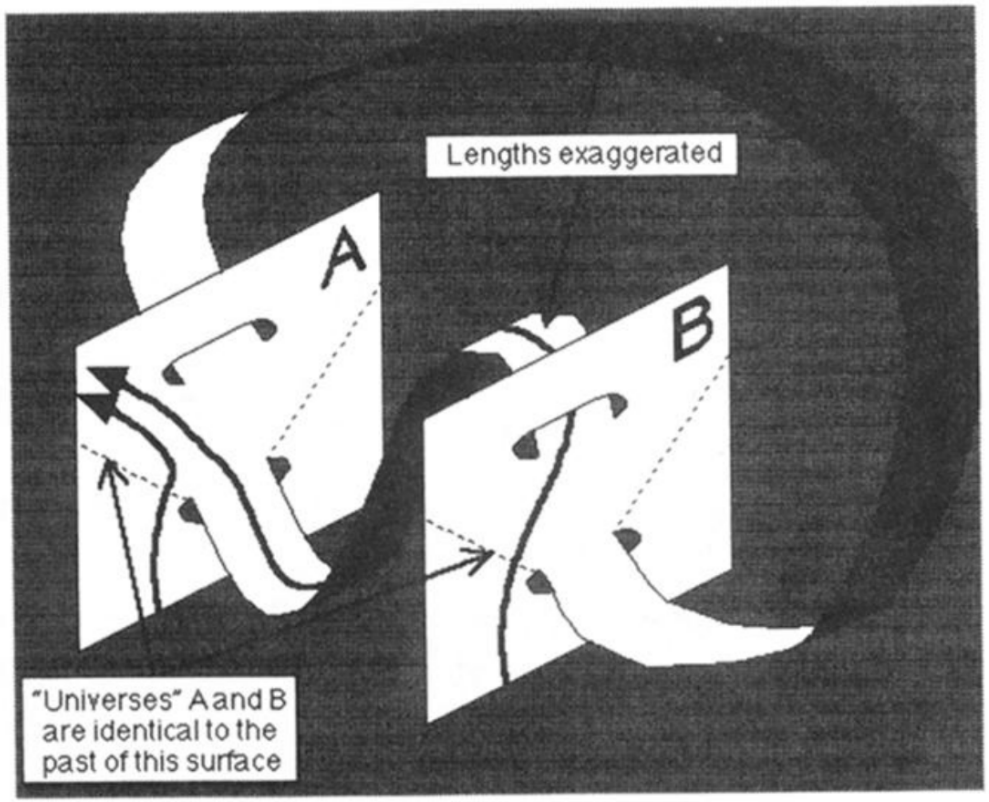

Figure 8: Deutsch's illustration of his solution of the grandfather paradox

Time loop interface

- Alter variables back in time
- fun setAtTime(time, variable, value)
- For example: setAtTime(0, result, 10)

Theorem

Variables in a program have a consistent probability distribution of values at every step in the program

Warming up: Factorizing numbers

- Factorizing the product of two primes is hard
- But O(1) with time loops

Algorithm 1 Factorizing N, which is a product of 2 primes

```
f := 2
t := currentTime()
if N mod f is not 0 or f >= N then
   setAtTime(t, f, f+1)
end if
return  f
```

NP

Definition

NP is the set of decision problems for which the problem instances, where the answer is "yes", have proofs verifiable in polynomial time by a deterministic Turing machine. (from Wikipedia)

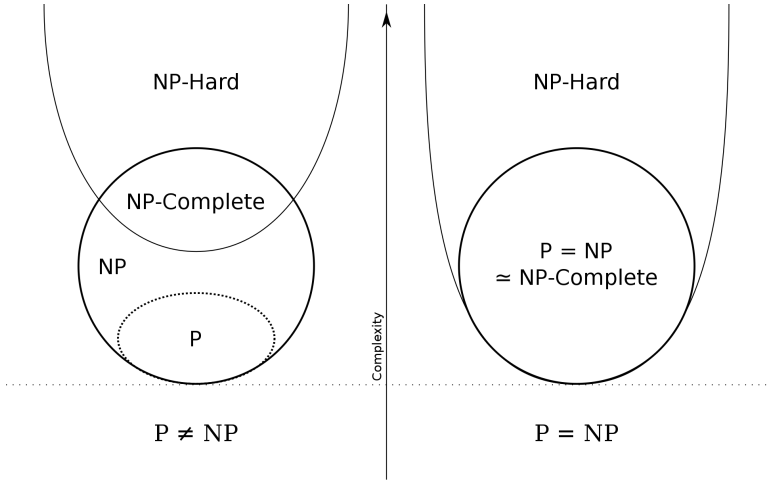

Figure 9: NP in relation to other complexity classes

Solving NP fast

- Goal: Solving NP problems in time loops of polynomial size

Algorithm 2 Paradox?

```
cert := firstCertificate
t := currentTime()
if cert is not a solution then
   setAtTime(t, cert, nextCertificate(cert))
end if
return cert
```

Solving NP fast, using probabilities

- The probability distribution needs to remain fixed

Algorithm 3 Using probabilities

```
cert := firstCertificate
t := currentTime()
if cert is not a solution then
   setAtTime(t, cert, nextCertificateLoops(cert))
end if
if cert is a solution then
   return true
else
   return false
end if
```

PSPACE

Definition

PSPACE is the set of all decision problems that can be solved by a Turing machine using a polynomial amount of space. (from Wikipedia)

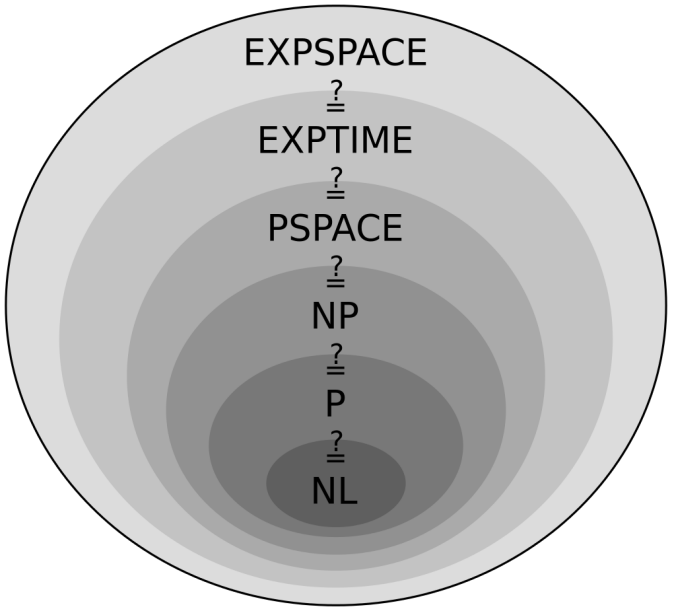

Figure 10: PSPACE in relation to other complexity classes

Solving PSPACE

Algorithm 4 Solving PSPACE given Turing machine T that decides

```
state := initial state of T
result := false
t := currentTime()
if state is accepting then
    setAtTime(t, result, true)
    setAtTime(t, state, initial state of T)
else if state is rejecting then
    setAtTime(t, result, false)
    setAtTime(t, state, initial state of T)
else
    setAtTime(t, state, nextState(state, T))
end if
return result
```

Can we do more?

- $PSPACE \subseteq P_{CTC}$
- But is that all?
- Can we do more in polynomial time loops?
- Not if we can simulate the time loops in *PSPACE*

What we will proof

- Proof by algorithm
- Any problem Q that is decidable in a polynomial time loop is in PSPACE because the time loop deciding Q can be simulated using the given algorithm

Lemma

Polynomially sized time loops can not decide any problem that is not in PSPACE

Induced Turing machine

- A time loop induces a deterministic Turing Machine
- After set at time, loop back to the given position with new values
- A loop in states gives probabilistic fix point

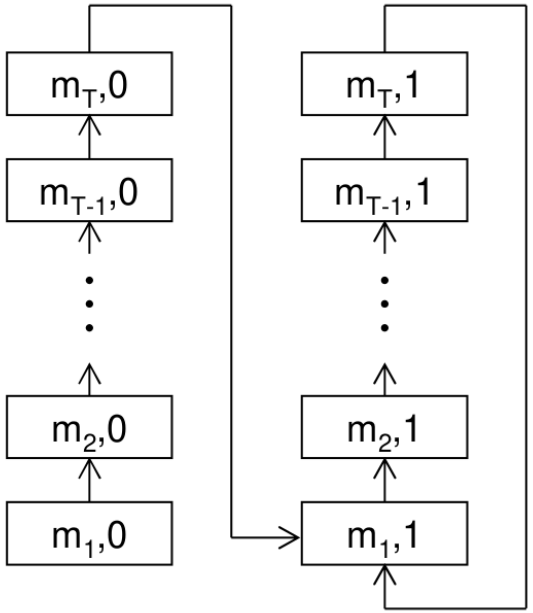

Figure 11: Example of an induced Turing machine from Aaronson and Watrous (2009)

Simulating time loop

- The state of the induced Turing machine is polynomially sized
- After simulating $2^{p(n)}$ steps we are in a loop
- This only uses polynomial space, so in *PSPACE*
- This state is a solution

Algorithm for simulating a time loop

Algorithm 5 Simulate a time loop in PSPACE

T := induced Turing machine of the time loop
state := initial state T
for $2^{p(n)}$ *steps* **do**
  state := step(T, state)
end for
return state

Lemma

Polynomially sized time loops can decide exactly the problems in PSPACE

Quantum computing

- Qubits
- Physically realistic
- Computational behaviour
- In time loops?

Figure 12: A quantum computer by IBM

Quantum computing in time loops

- Quantum circuit
- Same consistency condition
- Implementing classical algorithm possible
- Can compute PSPACE

Can we compute more?

- Aaronson and Watrous show we can't
- Complicated proof
- Show they can find a quantum operator that gives fix points
- Use it to find the fix point and result
- Can't compute more than PSPACE
- *"CTC's [time loops] make quantum and classical computing equivalent"* - Aaronson and Watrous (2009)

Conclusions

- Time loops might be possible
- Different models of computation might be possible
- In polynomial time loops, PSPACE is solvable
- Nothing more is solvable in polynomial time loops
- This is the same for quantum computing in time loops