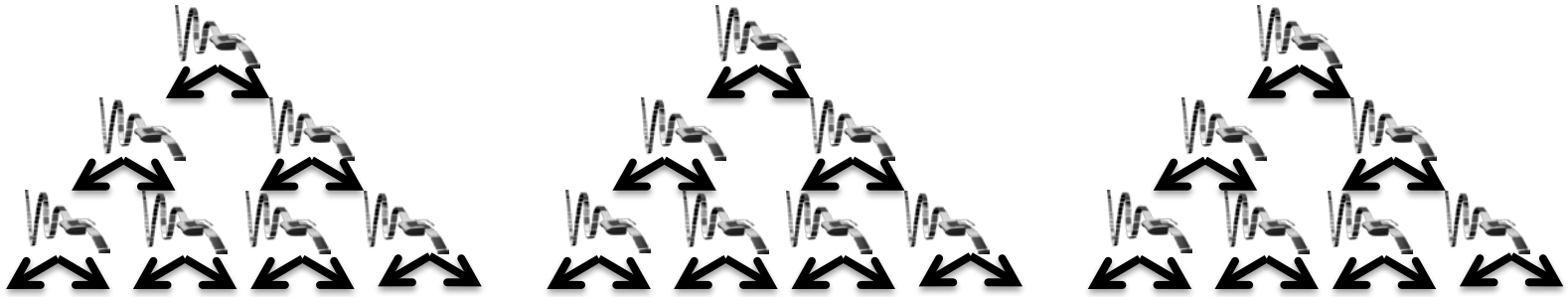
Computing Exponentially Faster:
Implementing a Universal Nondeterministic Turing Machine Using DNA

Ross D. King,
University of Manchester,
ross.king@manchester.ac.uk



Universal Turing Machines (UTMs)

- Universal Turing Machines (UTM) form the theoretical foundation of computer science.

- The Church-Turing *thesis*:
  - UTMs exactly define the concept of an algorithm - effective calculability.

- The Church-Turing *principle*:
  - UTMs are sufficient to perfectly simulate all physically realisable systems.

Universal Turing Machines

Alan Turing

Alonzo Church

UTMs: Maths or Physics?

- UTMs are a mathematical concept, but the language that Turing chose to describe them begs a physical interpretation.
- Digital electronic computers physically embody UTMs, but differ in that they have bounded memory, may only run for a bounded amount of time, make errors, etc.
- The tension between the mathematical and physical interpretation of computers is at the heart of computer science.
- Humans are ‘amphibians’.

Complexity Theory

- Given that a problem is computable (i.e. can be solved by a UTM in unbounded space and time), how much time and space is actually needed by a computation/algorithm?
- Informally, the complexity of an algorithm is its asymptotic worst-case use of a resource (space, time) as a function of the size of its input.
- I used to think that computability theory was much more profound than complexity theory, now I think the opposite

Complexity Theory

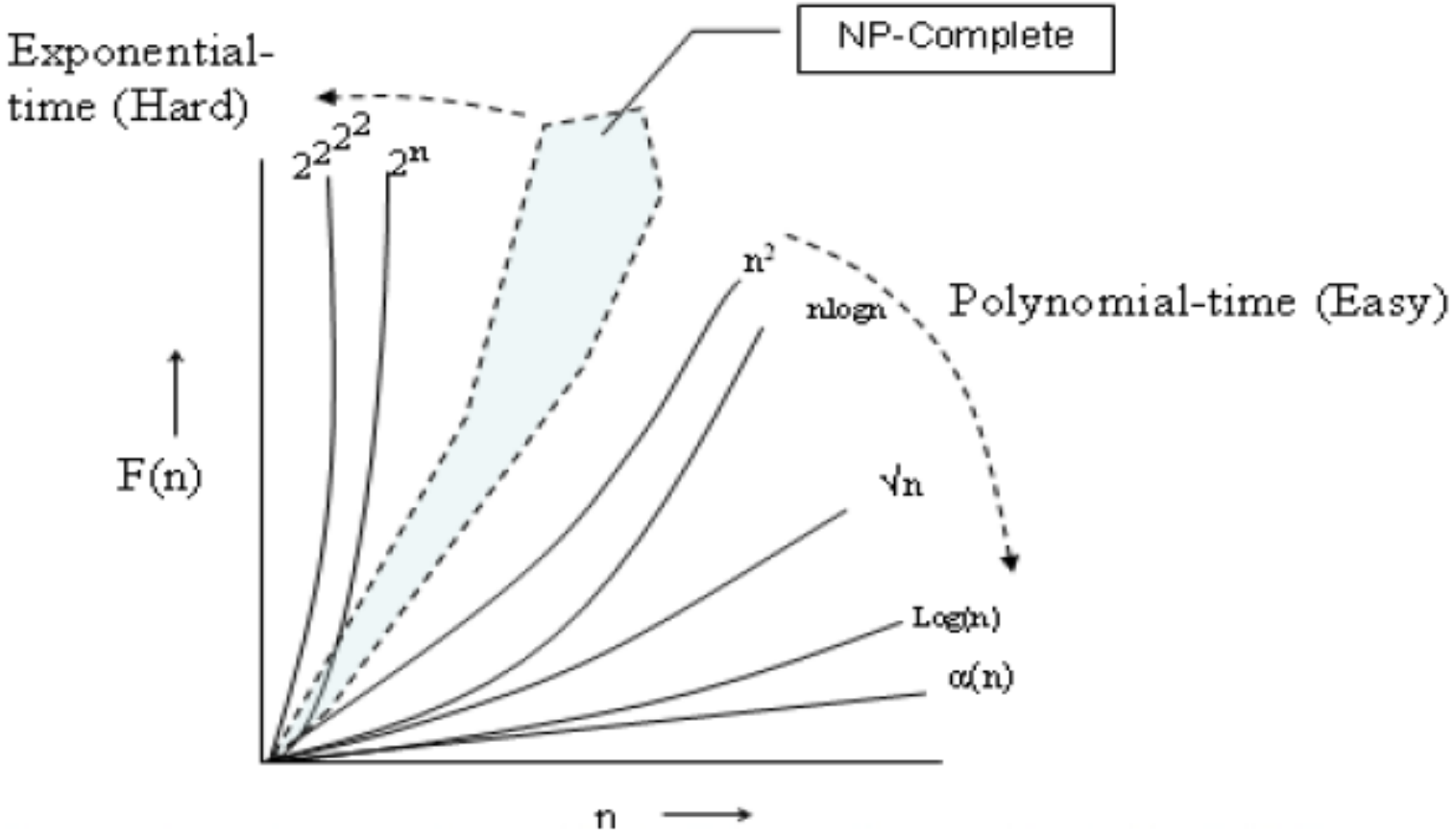
Stephen Cook

Richard Karp

The Feasibility Thesis

- The main conclusion of complexity theory is the ‘Feasibility Thesis’:
  - a natural problem has a feasible algorithm if and only if it has a polynomial-time (P time) algorithm.

Complexity Classes



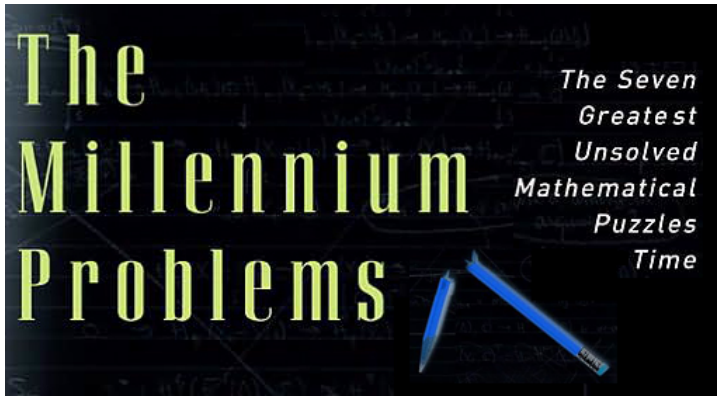
NP Problems

- The most interesting concept in complexity theory is the class of nondeterministic polynomial (NP) problems.
- Informally, this is the class of problem where the *solution* can be verified in P time.
- NP problems are the class that can be *solved using only* local interactions: only forks, no communication.
- NP complete problems are the most difficult in NP.

P ≠ NP ?

- The NP class is generally believed to be a strict superset of P, i.e. P ≠ NP. Intuitively this seems reasonable, as it would seem generally harder to find a solution to a problem than to verify a correct solution.

- However, this has never been proved, and the question whether P ≠ NP is the most important open problem in theoretical computer science, and one of the most important in mathematics.



P ≠ NP ?

"Although a practical algorithm for … P = NP would have devastating consequences for cryptography, it would also have stunning practical consequences of a more positive nature, and not just because of the efficient solutions to the many NP-hard problems important to industry. For example, it would transform mathematics by allowing a computer to find a formal proof of any theorem that has a proof of reasonable length, since formal proofs can easily be recognized in polynomial time." (Cook, 2000)

Maths v Physics (again)

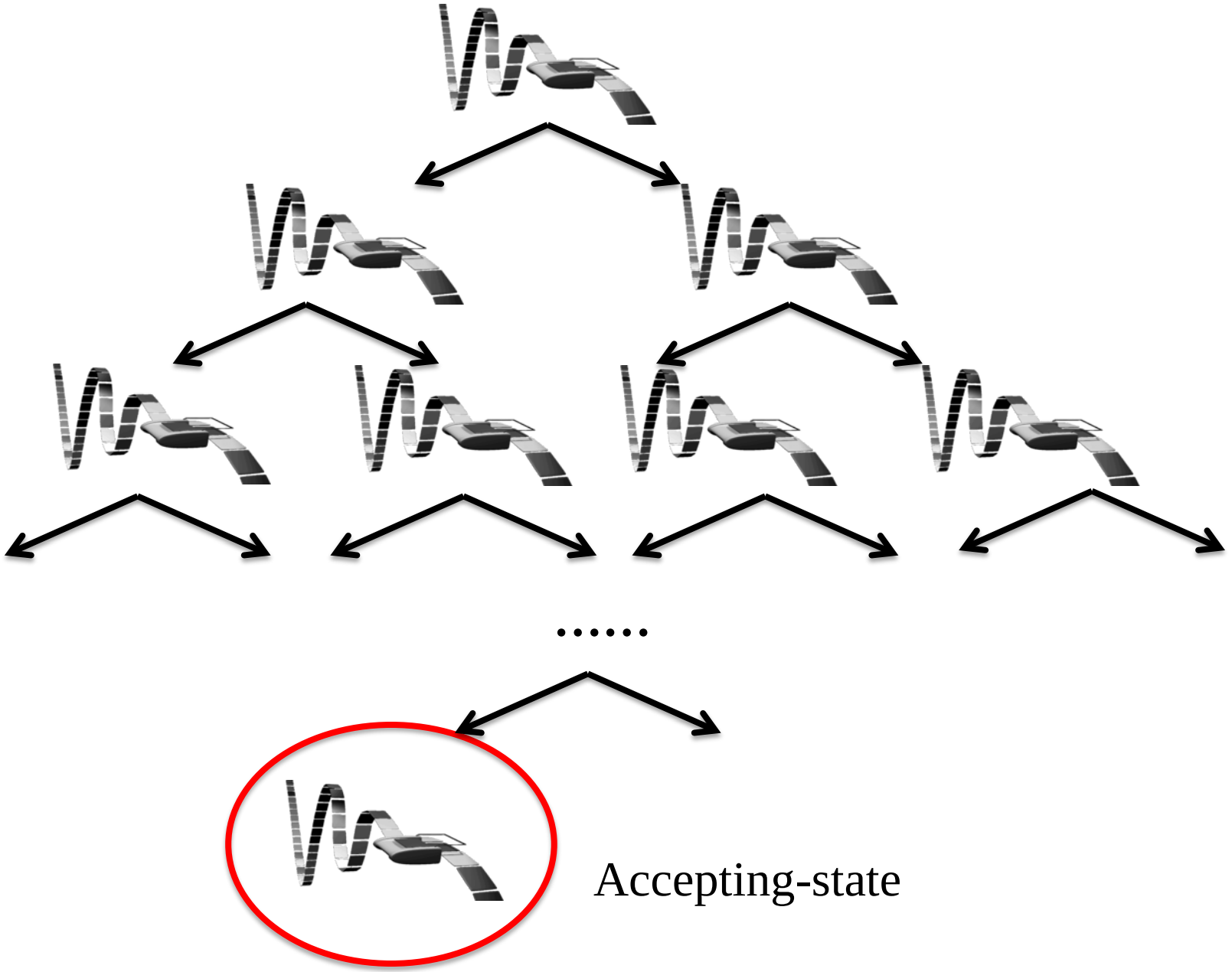
- It is important to distinguish the mathematical problem of the truth of the proposition ‘P = NP’, and the practical problem of solving NP problems in P time.
- The mathematical problem is constrained by a given set of axioms and proof methods, while all possible physical means may be used to solve the practical problem.
- Analogy: the classical Greek problem of trisecting an angle.

Nondeterministic UTMs (NUTMs)

- A now old-fashioned, but insightful, way of defining the class of NP problems is through the concept of a NUTM.
- NP problems are those that a NUTM can solve in P time.
- In a standard UTM: from a given state only one state can be be computed.
- In a NUTM: from a given state multiple states can be computed.

The Ontology of NUTMs

- There are two ontologically different ways of viewing how a NUTM efficiently solves a NP problem.
  - *Serendipity*. At each choice point the NUTM, in some unexplained way, chooses only the correct path. This explanation precludes any physical implementation, and one can read that NUTMs are ‘magical’ (Harel, 1992), ‘hypothetical’ (Reilly, 2004), ‘fictitious’ (Gowers, 2008), etc.
  - *Replication*. At each choice point the NUTM replicates and takes all pathways. This view opens up the possibility of physical implementation.


1
2
4
.....
2^n
Accepting-state

Thue Systems

Axel Thue

Emil Post

Thue Systems

- Formally a Thue system is the presentation of a Monoid.
- Informally:
  - A Thue system is a collection of rules of the form $w \leftrightarrow u$ where w, u are words in a finite alphabet.
  - A string e.g. $v\ w\ v'$ can be rewritten by the rule above to give $v\ u\ v'$.
- It is generally possible to translate any Turing machine into a Thue system, and vice-versa.

A Univeral Thue System

① a c ↔ c a

② a d ↔ d a

③ b c ↔ c b

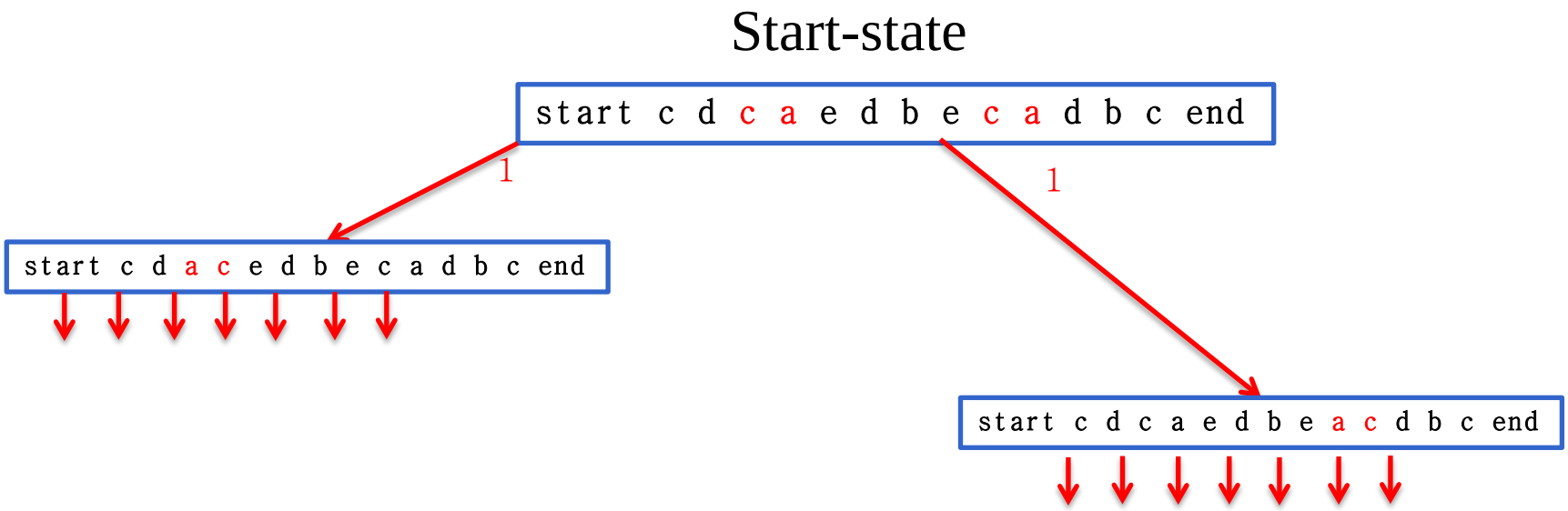
④ b d ↔ d b

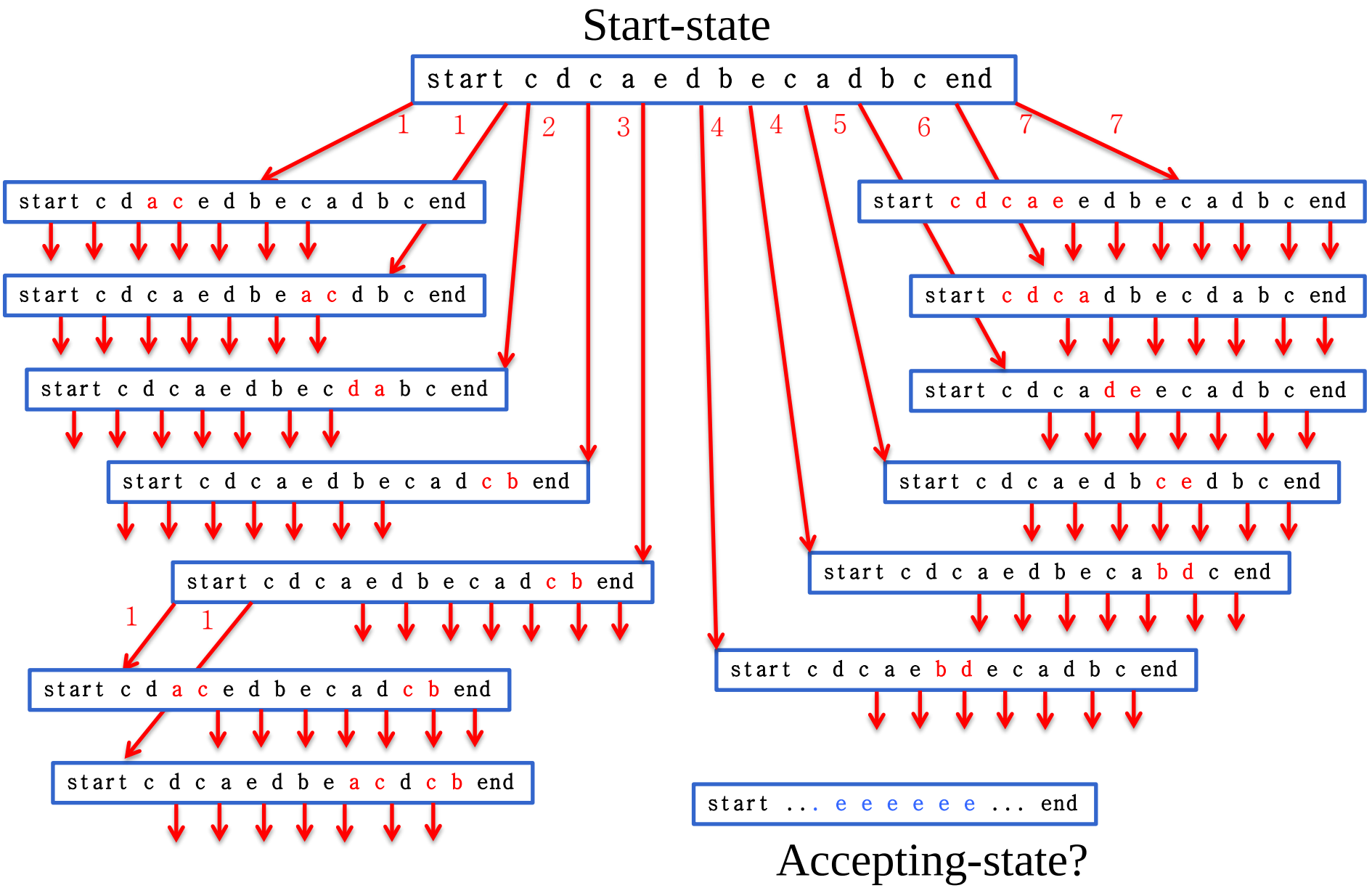
⑤ c e ↔ e c a

⑥ d e ↔ e d b

⑦ c d c a ↔ c d c a e

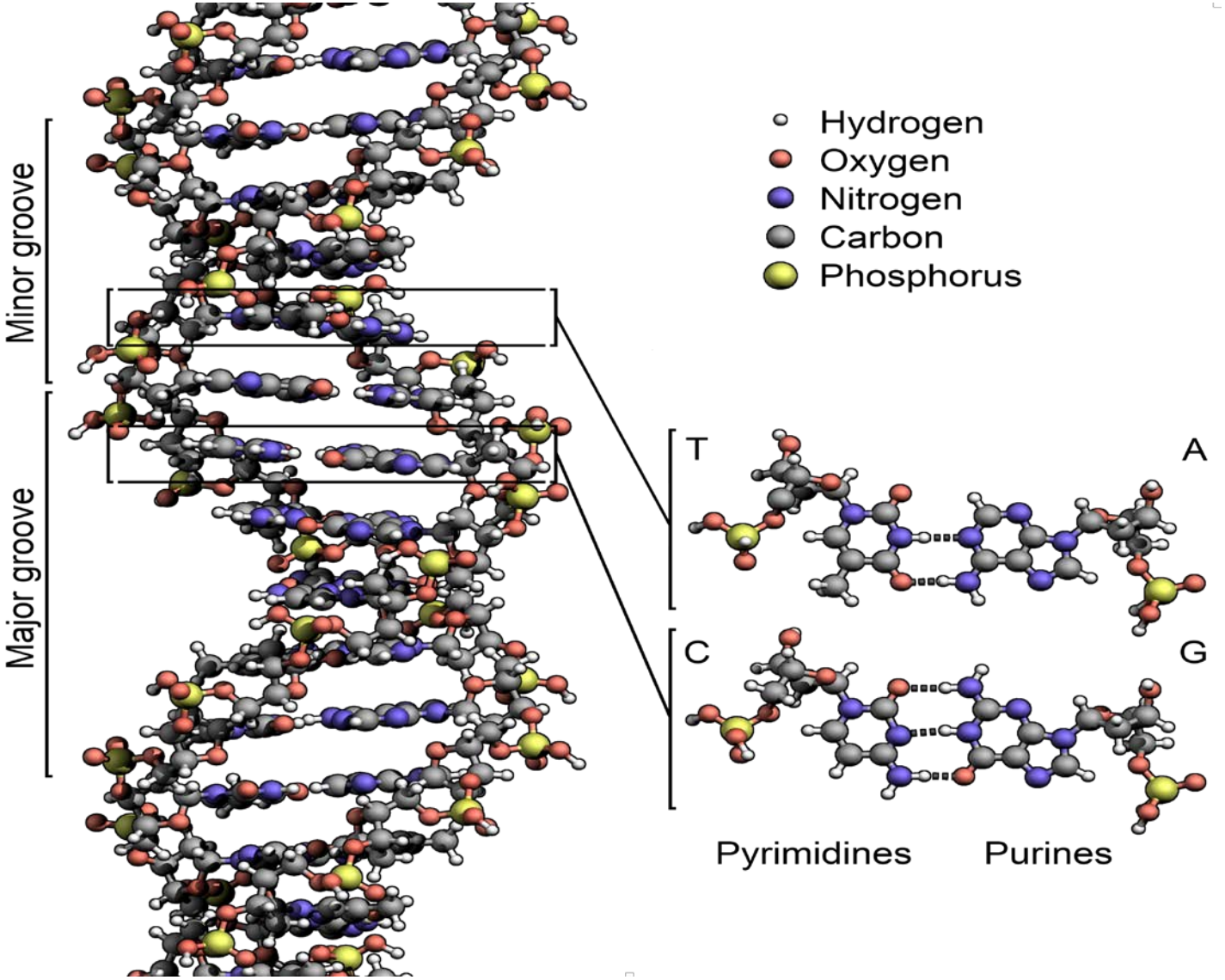
State Transition: ca →ac



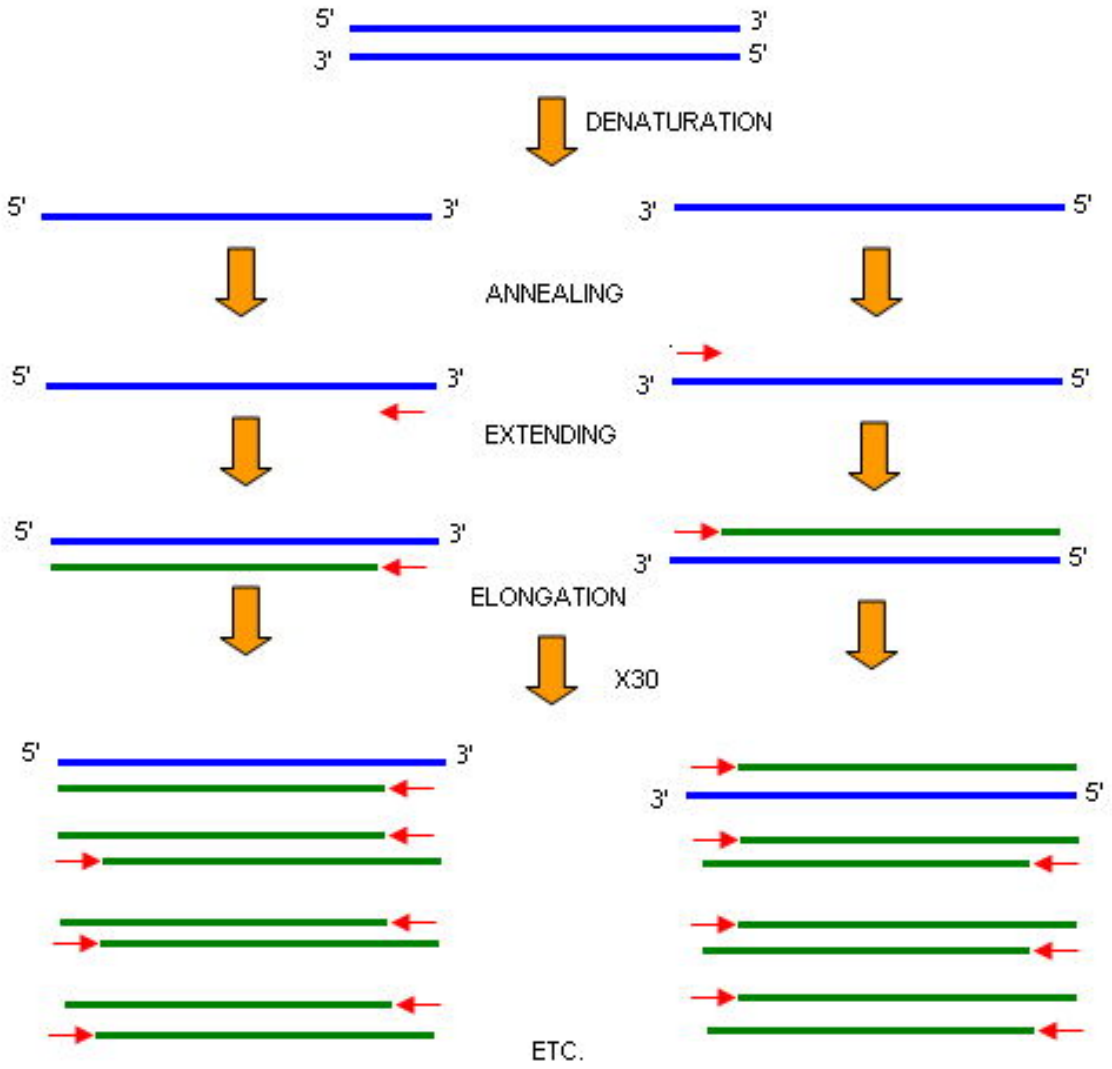

Start-state
start c d c a e d b e c a d b c end
1
1
2
3
4
4
5
6
7
7
start c d a c e d b e c a d b c end
start c d c a e d b e a c d b c end
start c d c a e d b e c d a b c end
start c d c a e d b e c a d c b end
start c d c a e d b e c a d c b end
start c d a c e d b e c a d c b end
start c d c a e d b e a c d c b end
start c d c a e e d b e c a d b c end
start c d c a d b e c d a b c end
start c d c a d e e c a d b c end
start c d c a e d b c e d b c end
start c d c a e d b e c a b d c end
start c d c a e b d e c a d b c end
start ... e e e e e e ... end
Accepting-state?

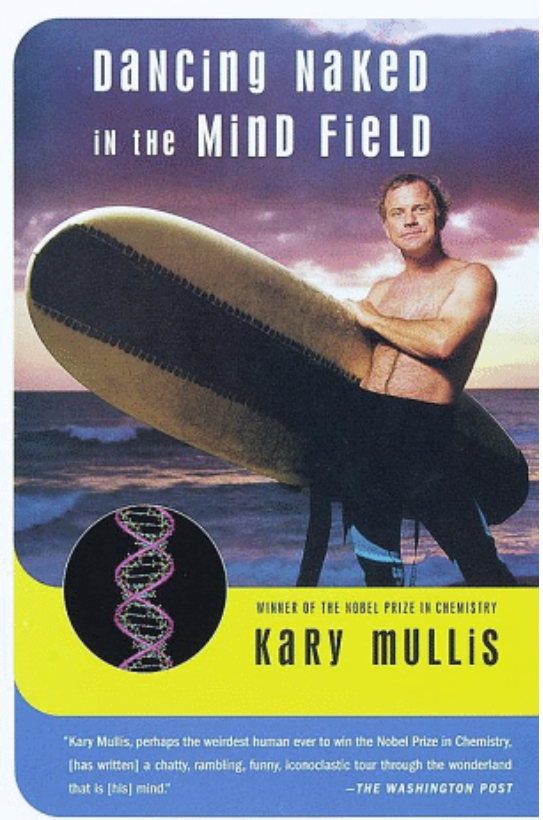
Molecular Biology

DNA


Minor groove
Major groove
Hydrogen
Oxygen
Nitrogen
Carbon
Phosphorus
T
A
C
G
Pyrimidines
Purines

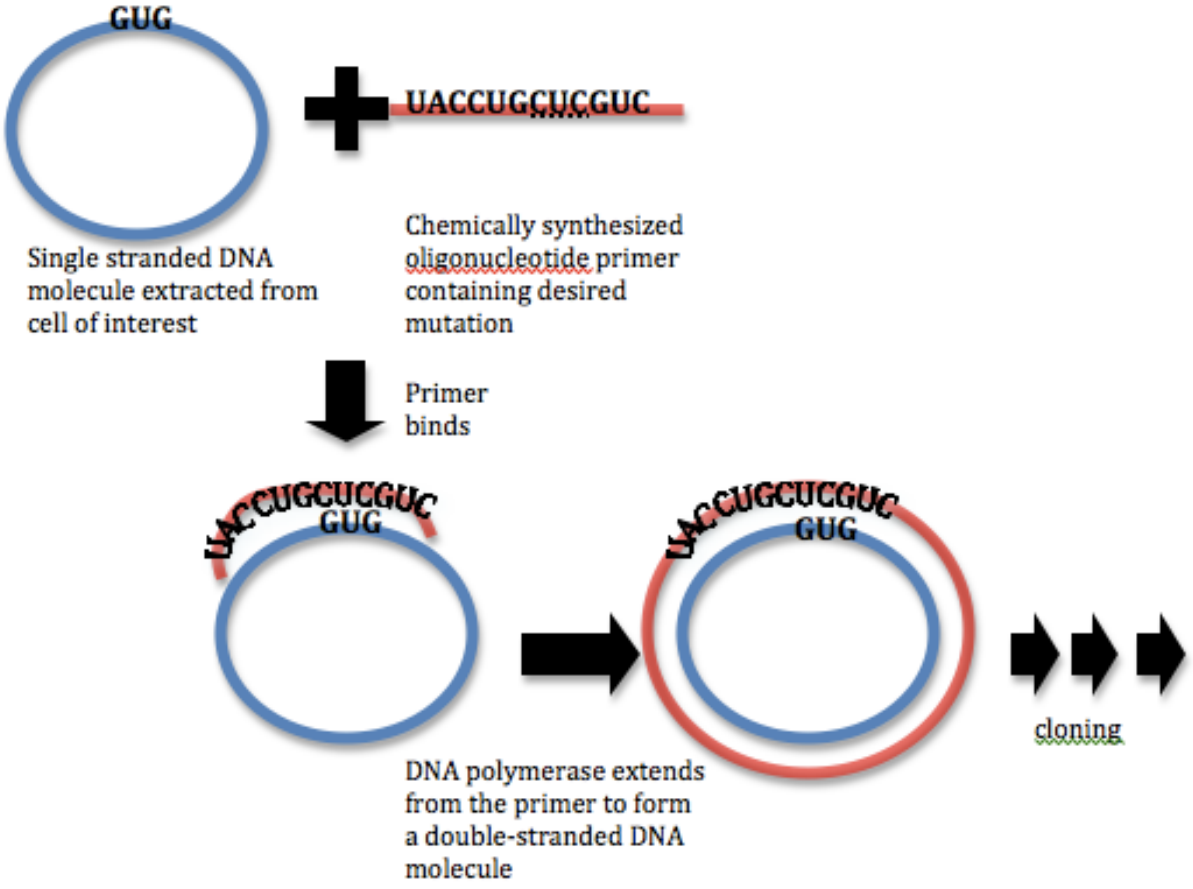
Polymerase Chain Reaction (PCR)





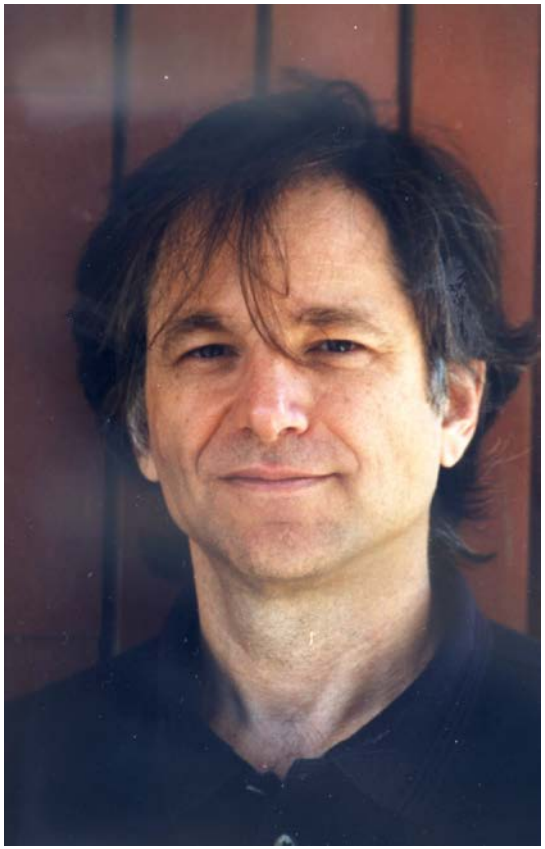
PCR

Site Directed Mutagenesis



Michael Smith

DNA Computing



Leonard Adleman

DNA Computing

- The number of operations per second could on a bench-top be ~10^{20}

  ~10^{4} x the fastest existing computer

- A DNA computer could execute ~2×10^{19} operations per Joule.

  ~10^{10} x more operations than existing computers

- DNA could enable an effective and persistent information density of ~1 bit per nm^3

  ~10^{9} x more dense than existing memory

Adelman's Model

- The foundational work on DNA computing was that of Leonard Adleman (Adleman, 1994).
- He demonstrated the solution of a seven-point Hamiltonian path by generating a set of random DNA sequences (possible solutions), and selecting a solution from the set.
- The significance of the choice of the Hamilton path problem is that it is a NP-complete one.

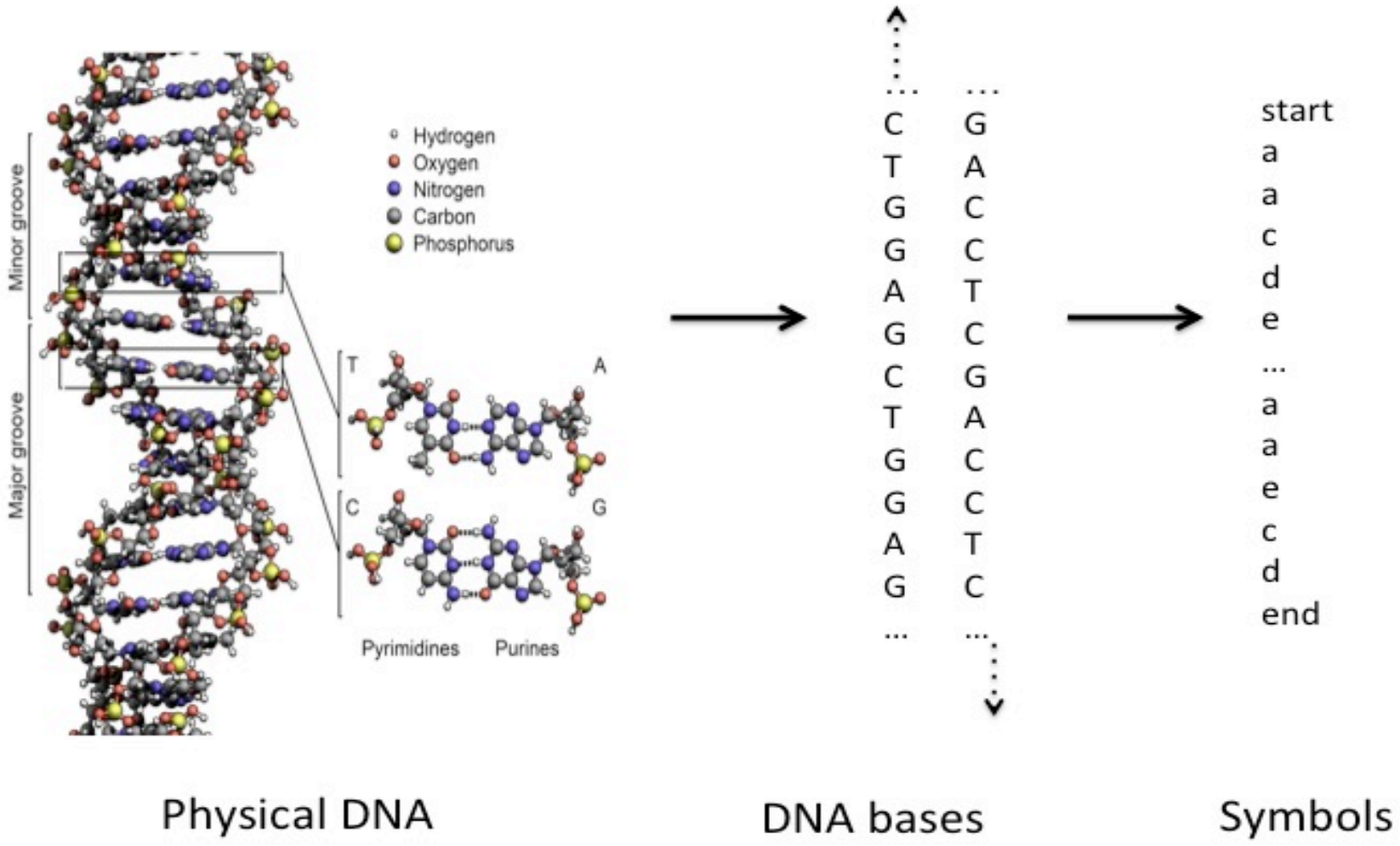
Building a NUTM using DNA

Design of a NUTM

- The initial program (start-state) of the UTM consists of a string (sequence) of DNA.
- During program execution all seven rewrite rules of the UTM Thue system are applied in parallel with no fixed order, and all possible combinations of rules are executed.
- The different DNA sequences produced correspond to different possible computational states.
- Execution halts when a particular sequence of DNA is produced, plus corresponding answers to the computation.

Symbols

Encoding of Symbols



Encoding of Symbols

- Sequences of the four nucleic acid bases (A, C, G, and T) encode information in quaternary code - as strings of bits (high/low voltage) encode information in binary code in an electronic computer.

- Require a balance between specificity (longer symbols) and differential PCR melting temperature (shorter symbols).

- Interestingly, 3 bases seems the best to encode symbols!

Symbols Used

Thue

a	TCT
b	GCT
c	TGG
d	ACG
e	CTG

Spacer

s	GAG GTG

Delimiters

start	TCGAAGGTCG
end	TAAGGATCCGGCTGCTAAC

Clamp

clamp	GGAATGTCAACATGATA

Intermediate

x1	CGG
X2	AAA

Writing Programs

- We need a way of writing arbitrary programs – sequences of Thue symbols.
- We used commercial gene synthesis to form our programs.

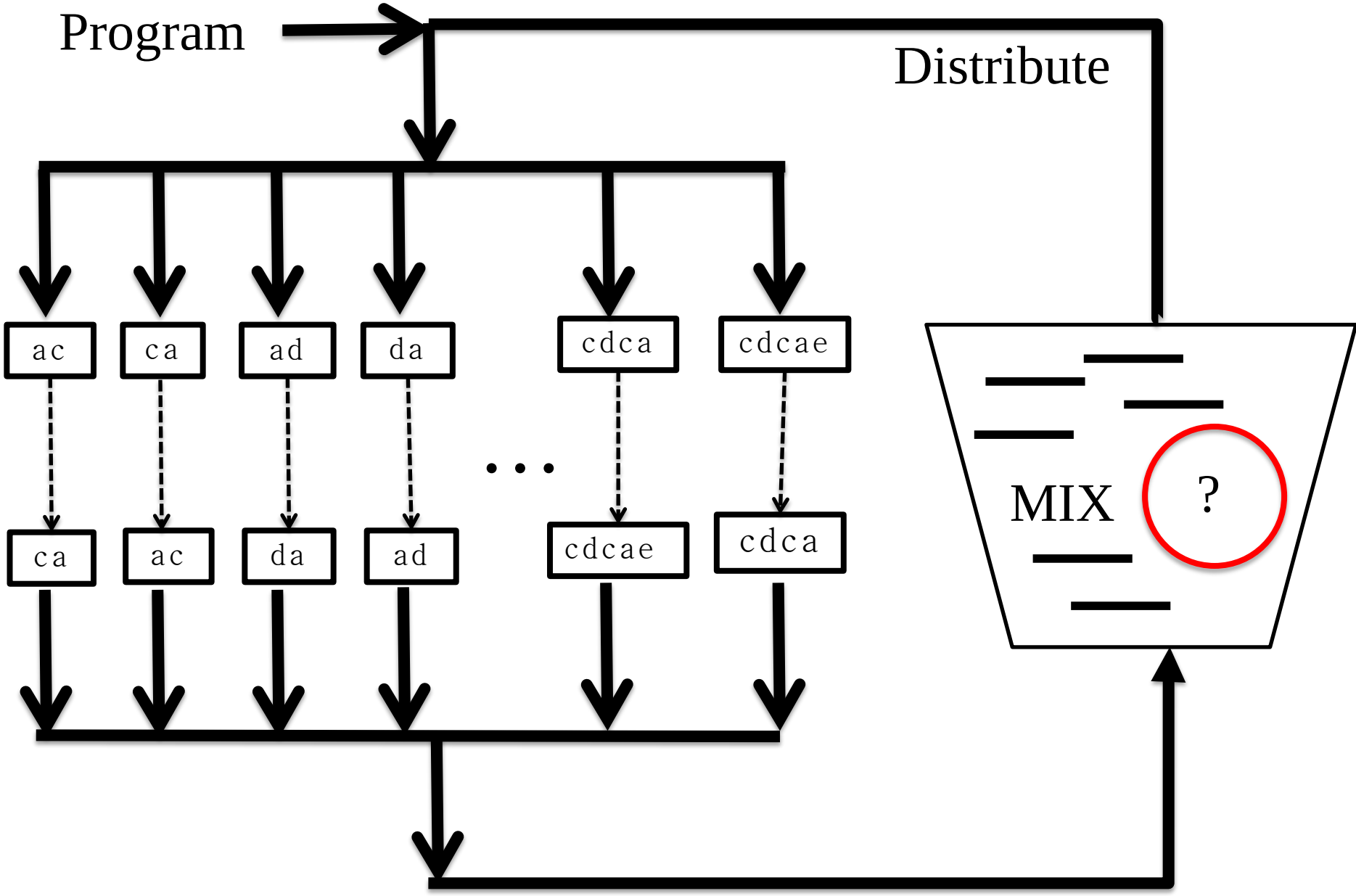
The Accepting State

- We need a way to recognise a specific state: the accepting state.
- Molecular biology has exquisitely sensitive ways of detecting specific sequences of DNA.
- We use sequencing, but in practice could use labeled complementary sequences.

Computer Architecture

Computer Architecture

- We need a design that:
  - Can recognise the halting state.
  - Apply all 7 (bi-directional) rules to all states.


Program
Distribute
ac
ca
ad
da
cdca
cdcae
ca
ac
da
ad
cdcae
cdca
MIX
?

Application of a Rule to all States

Rule Implementation

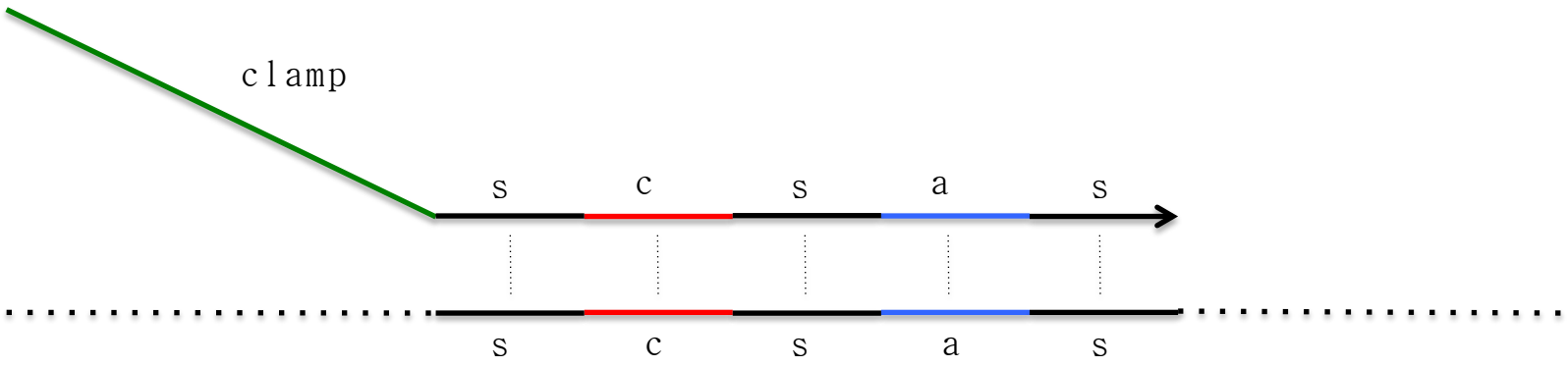
- It is convenient to separate the process into two parts.
- **Recognition**: identify all possible string rewrite positions for a Thue rule within a string of DNA.
  - String may contain an arbitrary complex pattern of Thue symbols, and then to insert a marker (clamp) sequence at the 5’ end of position. .
- **Rewriting**: execute the rule.

Recognition

Recognition

- Recognition is the more energetically challenging than rewriting.
- By this I mean the ΔΔG of binding is predicted to be lower between desired and undesired bindings.
- Recognition is achieved through use of a primer that binds the reverse complement of the Thue symbols (and spacers) in the head of the rewrite rule.
- This primer has concatenated to its 5’ end a marker sequence.

Recognition



The Boltzmann Distribution

$$\frac{F(State\ 1)}{F(State\ 2)} = e^{\frac{(\Delta 2 - \Delta 1)}{kT}}$$

- As this is an exponential, the probability of a much higher energy state being occupied is very low.

Recognition

- *In vitro* we have shown that we recognise and insert clamp sequences adjacent to the symbol strings **ba**, **ac**, and **ec** in a Thue string (template) containing multiple symbols combinations.

- For the **ba** symbol sequence, which occurs twice in the Thue string, as expected we inserted the clamp in the two correct places.

- Thus we have demonstrated the nondeterminism of rule recognition.

Simulation

- It is infeasible to empirically demonstrate that our recognition approach will work for all possible mismatching strings.
- We therefore use computational modelling.

- We have demonstrated that the probability that the correct symbols are recognised (bound) by the primer is higher than all possible mismatches.

Execution

Microprogramming

- The universal Thue rules are difficult to directly implement using existing molecular biology techniques.

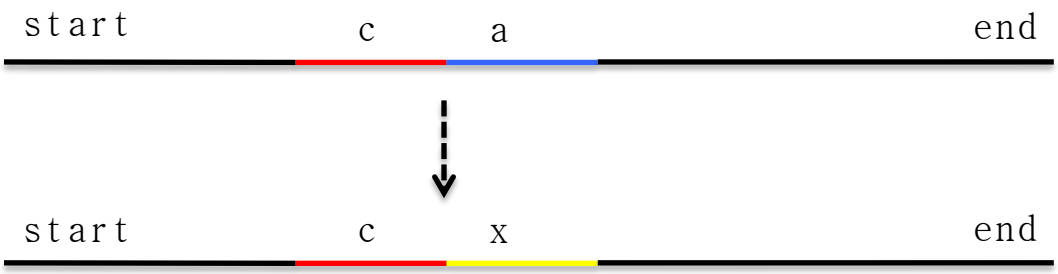
- We therefore developed an approach based on a few simple molecular biology operations that enable these rules (and others) to be implemented through combinations of these simpler operations.

- The approach is similar to microprogramming in CPUs.

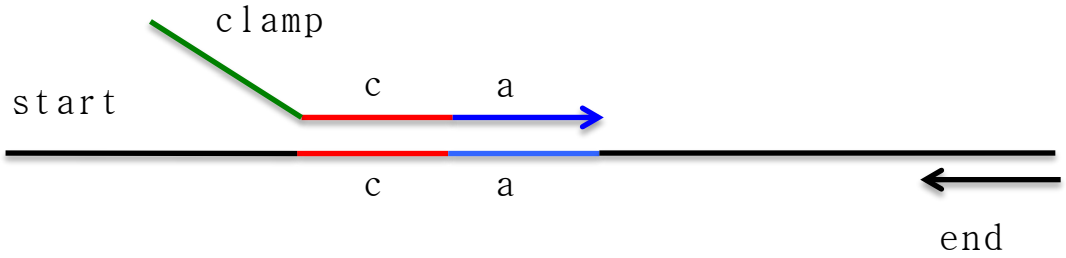
Microprogramming

- Example simple operations are:
  - Swapping a pair of Thue symbols
  - Changing an intermediate symbol to a Thue symbol
  - Insertion of an intermediate symbol.
  - Deletion of an intermediate symbol.

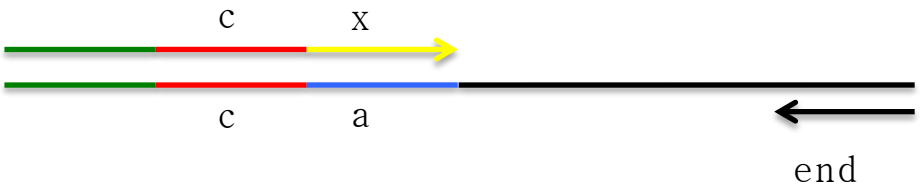
Changing a Symbol



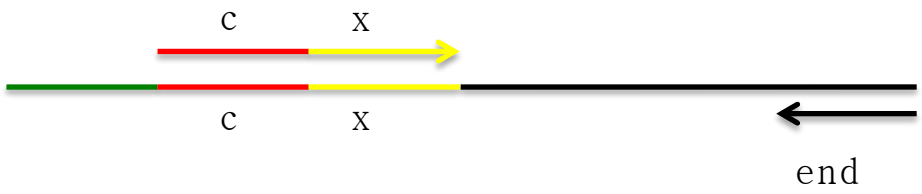
Recognition



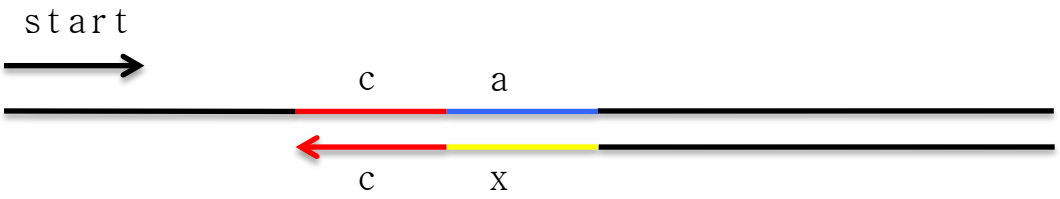
Change – strand 1



Removing the Clamp



Change – strand 2



Example Symbol Manipulation

1.

a)	**m R + E**	**primers**
	H R T	**template**
→	**H m R T**	**Insertion of clamp at rule head**
b)	**R + E**	**primers**
+	**H m R T**	**template**
→	**R T**	**Removal of clamp at rule head**

2.

a)	**m s_1 s_2 + E**	**primers**
	m s_1 s_2 T	**template**
→	**m s_1 s_2 T**	**activate pair clamp**
b)	**S + s_1 s_2 T**	**primers**
	H s_1 s_2 T	**template**
→	**H s_1 s_2 T**	**activate pair unification**

Symbol Interchange

Swap ce->ec:

Step	State	Operation	State(s) used
1)	HceT		
2)	mceT	1.a	1
3)	m**ce**T	2.a	2
4)	m**xx**T	8.a	3
5)	m**ec**T	9.a	4
6)	mecT	3.a	5
7)	**ce**T	1.b	3
8)	**xx**T	1.b	4
9)	**ec**T	1.b	5
10)	ecT	1.b	6
11)	H**ce**T	2.b	1, 7
12)	H**xx**T	8.b	11, 8
13)	H**ec**T	9.b	12, 9
14)	HecT	3.b	13, 10

Status of the Design

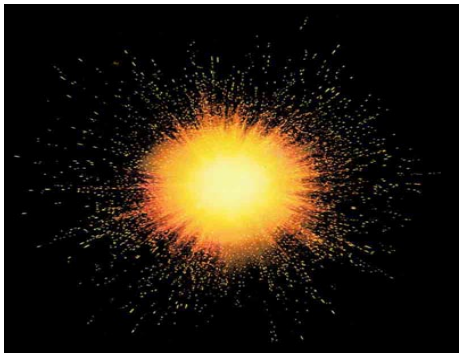
- We can swap, insert, and delete Thue symbols. This is sufficient to implement all the rules.
- Limitations:
  - Not all rules tested
  - Many controls to run
  - etc.
- No significant computation demonstrated.

Noise

- Our design for a NUTM approximates an abstract NUTM.
- This approximation is clearly less good than that for UTMs and electronic computers, mainly because of noise.
- Indeed the key to generally engineering practical computers is control of noise.
- Living systems demonstrate that noise can be controlled in DNA replication - many genes have remained virtually unchanged for a billion years.

Noise

- A number of approaches to noise control can be identified for DNA NUTMs.
  - Restriction enzymes could be used to eliminate any DNA sequences that do not encode symbols.
  - Error-correcting codes could be introduced to reduce the probability of incorrect rule applications.
  - Computations could be repeated multiple times (noise is unlikely to cause the same error to repeat); etc.
  - For NP problems the veracity of an answer can be efficiently tested by a standard computer (P time).



Limitations of NUTMs

- NUTMs solve NP problems in a brute force way by trading space for time: the amount of computation that occurs in each time step increases exponentially, but so does the amount of space used.
- The process resembles an explosion.
- Although DNA molecules are very small, they are still of finite size.
- This severely restricts the size of NP problem that a DNA NUTM could practically solve. Indeed the observable Universe has only ~10^{82} atoms.

Synthetic Biology

- A major motivation for this work was to develop a general-purpose way of controlling cells.
- The natural control of cells is a very complex combination of DNA, RNA, etc. This makes cells very difficult to reprogram.
- We argue that a more general approach is to engineer a DNA UTM, which would in principle enable any biological process to be programmed and executed.

Co-NP Machine

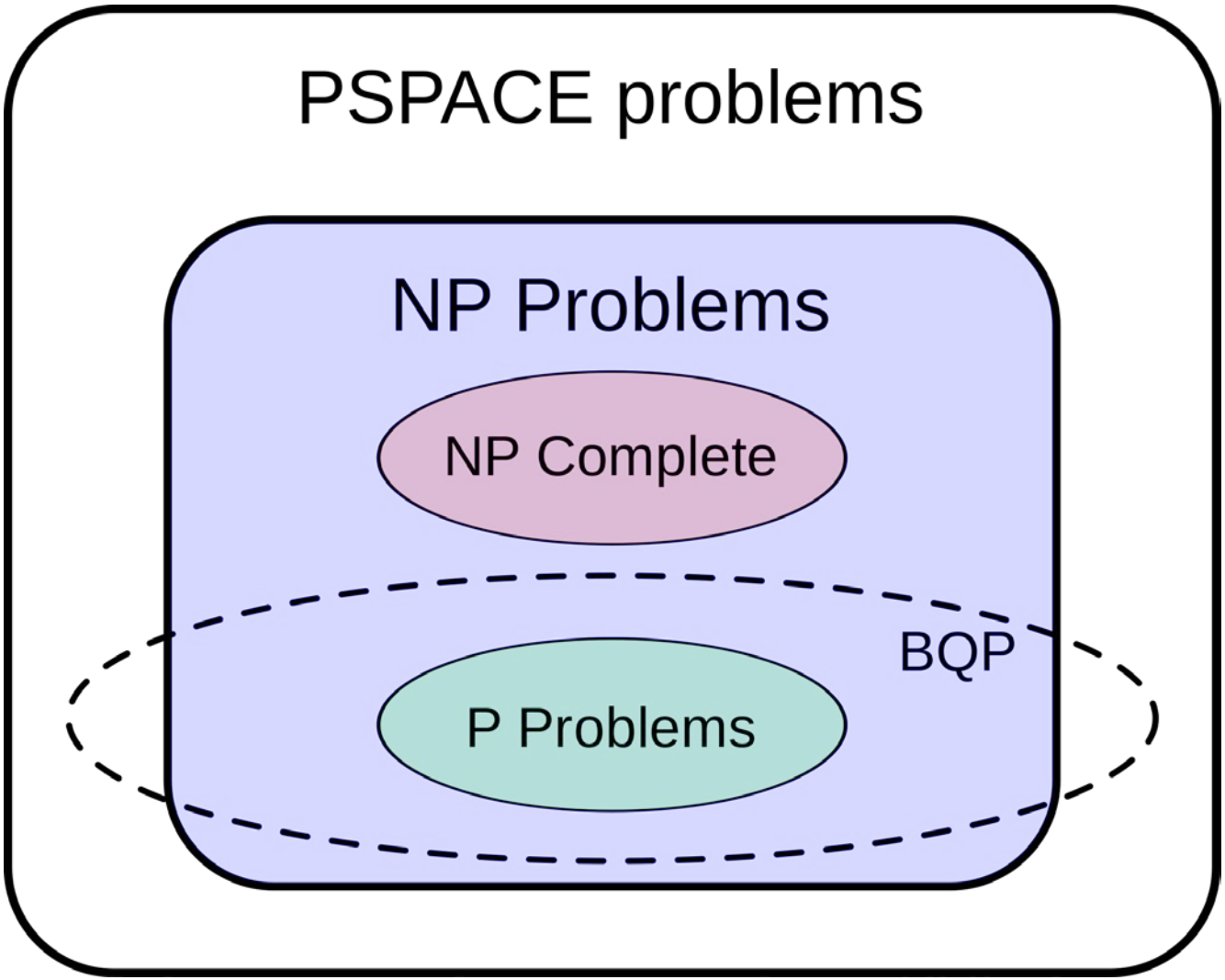
- A decision problem is a member of the class co-NP if and only if its complement is in the complexity class NP.
- The definition of NP uses an existential mode of computation:
  - if any branch of the computation tree leads to an accepting state, then the whole computation accepts.
- The definition of co-NP uses a universal mode of computation:
  - if all branches of the computation tree lead to an accepting state then the whole computation

- Should be physically feasible.

PSPACE Machine?

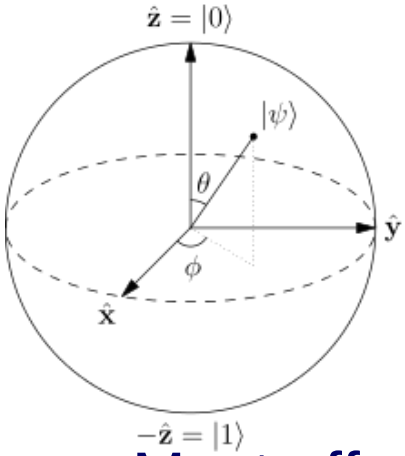
- Complexity theory treats time and space as fundamentally different: space is reusable while time is not.
- The resource limitation of the proposed physical NUTM is space, not time.
- The complexity class PSPACE includes those problems that can be solved using a Polynomial amount of space.

PSPACE



PSPACE v Physics

- In this regard it is important to note that in a NUTM all the computation is in a sense local: forks, no communication.
- It is therefore an important question of fundamental physics whether it is physically possible to build a computer that can efficiently solve PSPACE complete problems.
- According to physics the amount of computation that can be done in a given amount of space depends on the surface area, not volume: the holographic bound .



Quantum Computing

- Most effort on non-standard computation has focussed on developing quantum computers (QCs).
- Despite steady progress no Universal QC yet exists - Noise.
- The power of NUTMs and QCs seem orthogonal.
- Like NUTMs QCs utilize nondeterminism. QC's nondeterminism has the advantage of not occupying physical space (at least in our Universe), but is of a restricted kind.
- Combine them?

Is the Universe a NUTM?

- Advocates of the Many-Worlds interpretation of quantum mechanics argue that QCs work through exploitation of these multiple worlds.
- If the Universe were a NUTM it would explain the profligacy of worlds.
- Maybe the Universe is a Co-NUTM, and all paths will eventually be accepted.

Conclusions

- I have presented a new computational paradigm: Nondeterministic Universal Turing Machines (NUTMs).
- NUTMs can solve NP complete problems can be solved on P time.
- The current design has limitations, e.g. limited error-correction.
- The design opens up the prospect of engineering NUTM based computers grow to outperform all existing computers for certain classes of problem.

Acknowledgments

- University of Manchester
  - Funding
  - Colleagues: Konstantin Korovin, Andrew Currin, Maria Ababi, Katherine Roper, Philip Day, Douglas Kell.

- University of Cambridge
  - Steve Oliver.

- ERC Advanced Grant (ERC-2013-AdG 339781) – I would like to thank the ERC for **not** giving me this grant.

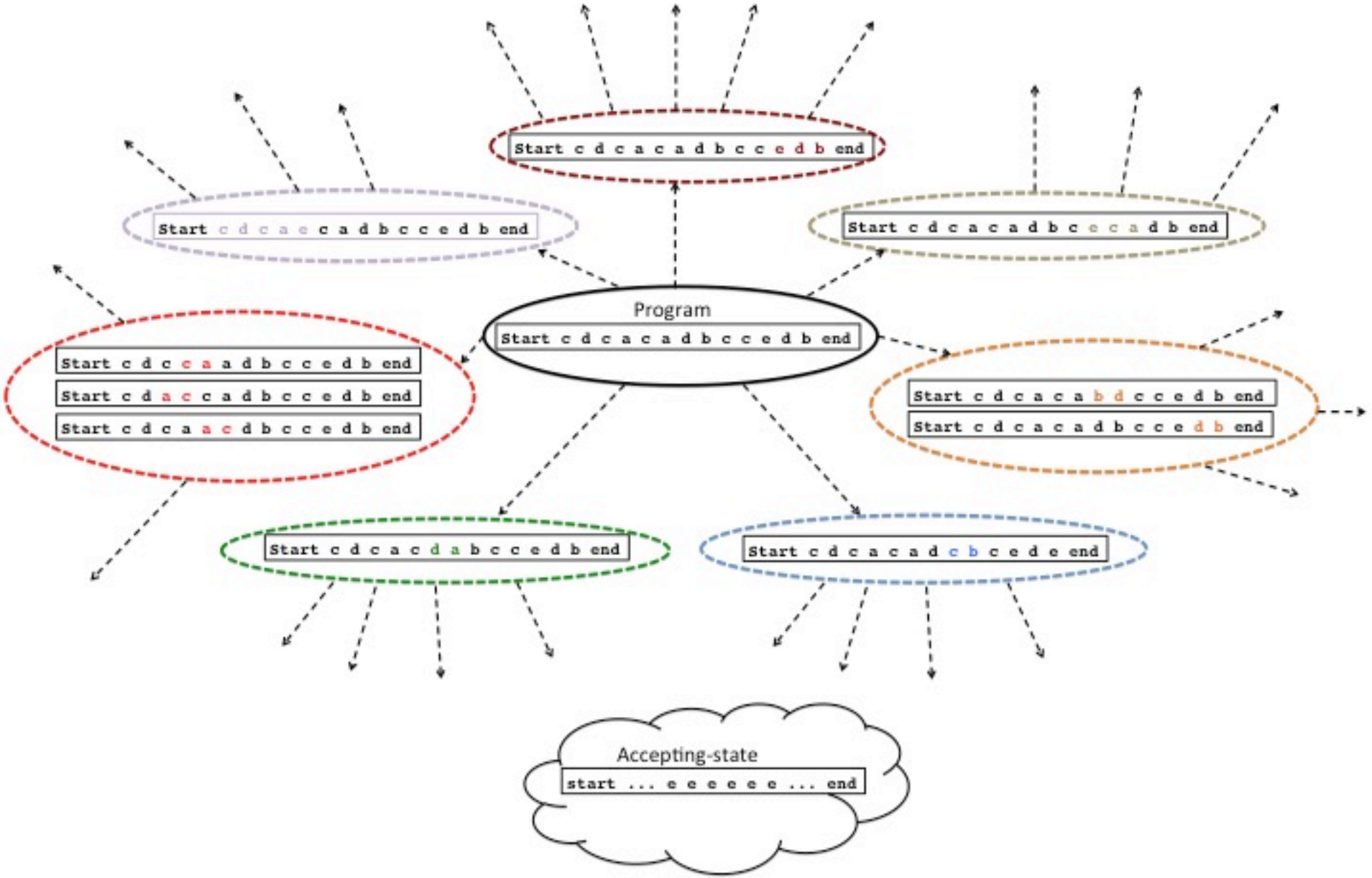
NP Problems

- Let C be a decision problem, then C is in the class NP if there is a function $V_c \in P$ and a constant k such that:

  If $x \in C$ then $\exists y$ with $|y| \leq |x|^k$ and $V_c(x,y) = 1$

  If $x \notin C$ then $\forall y$ we have $V_c(x,y) = 0$

- A sequence *y* which 'convinces' V_c that $x \in C$ is often called a 'witness' or 'certificate' (Wigderson, 2006).


Start c d c a c a d b c c e d b end
Start c d c a e c a d b c c e d b end
Start c d c a c a d b c e c a d b end
Program
Start c d c a c a d b c c e d b end
Start c d c c a a d b c c e d b end
Start c d a c c a d b c c e d b end
Start c d c a a c d b c c e d b end
Start c d c a c a b d c c e d b end
Start c d c a c a d b c c e d b end
Start c d c a c d a b c c e d b end
Start c d c a c a d c b c e d e end
Accepting-state
start ... e e e e e e ... end