Babbage Session Two: Minutes/Synopsis

Babbage:
Ben, Bernard, E'beth, Ed, Han, Rich, Yael
with a special appearance by Neil Gershenfeld.
Transcript by Rich.

June 8, 1998

Abstract

This paper is a synopsis of second Babbage meeting, being partly meeting minutes and partly a digest of the material presented. As closely as possible I have tried to stick with my original transcription, but the complexity of the material and my inability to typeset all of the tables and equations properly in "real time" has necessitated revisions and corrections based on Hopcroft and Ullman. Since future presentations should be accompanied by online presentation material, detailed technical transcripts should be unnecessary, except where new technical material comes up in the course of digressions and discussions.

I encourage all participants to review my work and make additions and corrections to the LaTeX source which I will post along with the HTML and PostScript versions of this document. — Rich.

Contents

1 Finite Automata and Regular Expressions — Bern

Define FA:

1. discrete inputs and outputs
2. finite number or internal states

1.1 Deterministic FAs

Deterministic FA – define by 5-tuple $(Q, \Sigma, \delta, q_0, F)$

- Q — finite set of states
- Σ — input alphabet
- δ — transition function mapping $Q \times \Sigma \to q$
- F — set of final states

A string x is accepted if $\delta(q_0, x) = q \in F$. A language $L(M)$ is defined as $L(M) = \{x|\delta(q_o, x) \in F\}$

Critical notion is that for deterministic FA there is a one-to-one mapping between $\Sigma \times Q$ and Q.

Example of deterministic FA: man, wolf, goat problem[1].

Notion that allowed states are determined by transitions.

Another example is the FA which accepts only zeros and ones.

Question regarding infinite vs. finite strings and bearing on FA. Is an infinite string a legal input to FA? Strings are of any length but must be finite.

1.2 Non-deterministic FAs

A non-deterministic FA is an FA with more than one mapping for a single state-input pair.

An example of a non-deterministic FA is presented which accepts strings with either two consecutive zeros or two consecutive ones[2].

Question as to which transition to make in response to input. Automata forks to produce parallel automata parsing input. Input is accepted if any parallel automata accepts. Otherwise reject.

This following table describes the non-deterministic FA:

δ	0	1
q_0	$\{q_0, q_1\}$	q_1
q_1	$\emptyset$	$\{q_0, q_1\}$

(1)

These equations and the following table describe the deterministic equivalent:

$$M' = (Q', \{0, 1\}, \delta', [q_0], F') \quad (2)$$

$$Q' = \{[q_0], [q_1], [q_0, q_1], \emptyset\} \quad (3)$$

[1] See Hopcroft and Ullman, pp. 14–15.

[2] This description does not match the language accepted by this FA; A transcription error on my part, no doubt. — Rich.

δ'	0	1
$[q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_1]$	$\emptyset$	$[q_0, q_1]$
$\emptyset$	$\emptyset$	$\emptyset$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

(4)

1.3 FA with ϵ moves

FA with ϵ moves mentioned – silent transitions on no input. These FAs are provably equivalent to NFAs[3].

1.4 Two-way FA

A two-way FA allows movement backward and forward in input string. These are provably equivalent to one-way FAs[4].

1.5 FAs with output

1.5.1 Moore machine

A Moore machine is a discrete finite automaton with output. A Moore machine is specified in the same way as previous FAs, but with the addition of Δ and λ:

- Δ is the output alphabet
- $\lambda(Q) \to \Delta$ (λ maps states to outputs)

1.5.2 Mealy machine

A Mealy machine is an FA which also produces output, but does so on transitions. In a Mealy machine $(Q \times \Sigma) \to \Delta$ (transitions map to outputs).

1.5.3 HMM and NMM digression

HMM and NMM (Markov models) — similar to FA with output, but state transitions probabilistic. Used in speech recognition, etc. (Hidden Markov models and Markov models) Bays (sp?) nets also mentioned.

2 Properties of Regular Sets — E'beth

1. Pumping Lemma
2. Closure Props
3. Myhill/Nerode
4. algorithms for other ???

[3] Hopcroft and Ullman, pp. 26–27
[4] Hopcroft and Ullman, pp. 40–41.

2.1 digression on nature of class

What questions do we want to answer — what is computation? What do different architectures mean? Can we come up with a fundamental definition of computation.

Class is not limited to Turing machines or analogs, but it makes sense to start there.

Computation as algorithm

Computation as tool for reducing uncertainty (Han). Information theoretical take on computation. Is signal processing computation?

Information manipulation can be computation – what does it take? Are linear operations computation? Is there a computation phase transition?

Tie physics to operations as a way of narrowing computation – deal with non-infinite resources. (Ben)

Qbits mentioned – Qbit annihilation.

2.2 Pumping Lemma

2.2.1 Regular Sets

A regular set is a language accepted by an FA. An important point is that not all strings are members of a regular set; Prime numbers represented as binary strings are an example of a non-regular set.

Ben's prime number joke A man says he has come up with an algorithm for generating prime numbers — all odd numbers are prime. Three "experts" are called in to verify this. All verify for different reasons:

mathematician 1, 3, 5, and 7 are prime, the rest by induction.

physicist 1, 3, 5, 7, 9 – experimental error, 11.

engineer All odd numbers are prime.

2.2.2 Regular Sets Continued

The idea is that for all strings accepted by an FA of length greater than the number of FA states, there must be repeated substrings.

More formally, if n is the number of states in the smallest FA which accepts L, if the FA accepts strings of length $m > n$, there must be repeated substrings[5].

Ben's synopsis Consider a finite state machine with ten states; god gives me a string which is 20 symbols long which results in an accept. Since there can't be a unique state for every symbol in the string, it means that I have repeated at least one state; There must be a state loop in the machine.

[5] See Hopcroft and Ullman, p. 55, theorem 3.1

This means that one can construct a string of arbitrary length that will be accepted by the FA by concatenating multiple "loop" substrings at the appropriate point in an otherwise accepted input string.

Han's elaboration Regular languages are a subset of Σ^*. A regular language is accepted by a FA, and FAs must have a finite number of states. Therefor, if I have a regular language there is a FA with a minimum number of states n.

2.3 Neil Gershenfeld's digression

Enter Neil Gershenfeld, stage left.

Neil asks what the structure of the class is. We tell him we're making it up as we go along. Neil says that the language of computation is wedded to classical notions of computing architecture, but its the only language we have right now. It would be nice to write a text book, nice to put together cook book for computational universality. Babbage could end up producing a series of papers like the chaos collective.

Need to learn coding theory (Blahat – information theory and coding theory). Coding theory at least as important as computation (Steve Omohundro has written papers on: do PDEs compute?)

Need to involve Deepa in class[6].

One of Neil's favorite questions: When will we want to start making machines which signal like neurons? Jerry Lettvin has made a career of studying this kind of thing (Homework problem of contact him and invite him to the group) Neil describes him as an imposing person who has spent lifetime thinking about engineering and biology and signaling. Gil Pratt (Lettvin's mentor) has also done lots of work in neuron-like signaling.

What is the menu of questions that you would ask to come to the conclusion that you should signal like neuron?

(Ben) little knowledge of brain regime. (Neil) lots known about neurons. Neuron like signaling should allow construction of messy computer – low tolerances on components.

Al Bar and Tad Hogg – noisy gradient descent. (question about gradient descent and Q-bits)

Presume that we will write book on subject. Start with this assumption and use it as organizational basis.

Exit Gershenfeld

2.4 Han's explanation of FA limitations

$$0^n1^n2^n = 012, 001122, 000111222, \ldots$$

This example breaks regular language, because number of each element is critical. This prevents pumping (no way to construct larger strings using a single repeated substring) , and thus you would need an arbitrarily large FA.

[6] Deepa is interested in neuron-like signaling and its implications for computation.

Connection between stacks and CFGs – the addition of the stack to a FA results makes a pushdown automata. The infinite stack allows a the acceptance of a context free languages (such as the example above) by pushdown automatas which cannot be accepted by FAs.

2.5 E'beth's presentation continued

Closure of regular sets[7] – $L^* = \bigcup_{i=0}^{\infty}$

$$L \subset \Sigma^* \tag{5}$$
$$\bar{L} = \Sigma^* - L \tag{6}$$

Decision algorithms for regular sets

2.6 digression on class structure

Can presentations be made more efficient? Put everything on the web by Sunday to make sure people have time read before class.

3 Presentation three — Yael

(insert Yael's typed notes here)

4 Ben's presentation

4.1 Properties of context-free languages

The pumping lemma for context-free languages is not as powerful as the pumping lemma for FAs. Once you move into CFGs, you run into the halting problem. You can't tell if a CFG will accept a given language, as you can with an FA.

4.1.1 halting digression

Finiteness is intimately related to halting, somehow. Multiple stacks in a pushdown automata make it equivalent to a Turing Machine.

4.2 Turing Machines

A Turing machine is composed of an infinite tape bounded on the left, a read-write tape head, and a finite control.

[7] This is a particularly poorly documented section; My apologies to E'beth. The discussion of closure properties is section 3.2 of Hopcroft and Ullman (pp. 58–63) and decision algorithms is section 3.3 (pp. 63–65)

4.2.1 Black Box digression

Each level in the Chomsky hierarchy more capable than previous level. Black box generating language is acceptable by which kind of machine tells about contents of black box.

4.3 TM's continued

A Turing Machine (TM) is described by[8]

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)\,, \tag{7}$$

where

Q is the finite set of states,

Γ is the the finite set of allowable tape symbols,

B is the blank symbol ($B \in \Gamma$).

Σ is the set of input symbols, $\Sigma \subset \Gamma, B \notin \Sigma$.

δ is the next move function, $\delta(Q \times \Gamma) \to Q \times \Gamma \times \{L, R\}$.

q_0 is the start state ($q_0 \in Q$).

F is the set of final states, $F \subseteq Q$.

For each symbol scanned, a TM:

1. changes state,
2. prints a symbol on tape cell, replacing what was written there before, and
3. moves the tape head left or right one cell.

instantaneous description of Turing machine All symbols to left of head, State of machine, symbol head is scanning and all symbols to right of head, *i.e.*

$$(X_1 X_2 \ldots X_{i-1} q X_i \ldots X_n).$$

Example of Turing machine accepting a string with equal numbers of zeros and ones – this can't be done with FA, as was previous shown.

Programming Turing machine can be done entirely in finite state logic, but can also be done with information on tape.

Finite state logic can also be used to store information, by including tape symbol dependent states.

[8] Hopcroft and Ullman, Section 7.2, p. 148.

4.4 Two-way infinite tape

A two way infinite tape TM can move indefinitely in either direction. This can be shown to be equivalent to a one way infinite tape TM by the following argument: It is clear that two one-way infinite tape TMs can be used to simulate a two-way infinite tape. If these are lined up "side by side" and the two tapes are interleaved, then a single one-way infinite TM can be constructed which is equivalent.

4.5 Multiple tape TM

Multiple heads, multiple tapes, one control logic. A similar interleaving argument can be used to multiplex these tracks on a single track if the heads are tied together. If they are not tied together, then shifting can be used to move information around to avoid head collisions.

$$\begin{array}{ccccc} \text{head one region} & & \text{head two} & & \text{head three} \\ \text{xxxxxxxxxxx} & \ldots & \text{yyyyyyyyyyyyy} & \ldots & \text{zzzzzzzzzzzzzz} \end{array} \tag{8}$$

4.6 Multidimensional TMs, Non-deterministic TMs, *etc.*

Many Turing constructions are possible, and virtually all are equivalent to the classic single-tape deterministic one-way TM.

4.7 Church Turing hypothesis

Partial recursive functions are the only computable functions, and these are the functions computable by TMs.

4.7.1 Question about functions and computability (Yael)

There are uncountably infinite number of functions, and only a countable infinite number of procedures. This means that there are some functions which can't be computed. This proposition hinges on notion that procedures are finite.

5 Discussion for next time

What will we cover – NP complete and Ed and Rich's discussion, or go more fundamental. Ben wants more in-depth computability theory, Yael wants to examine basic/fundamental questions about computation.

5.1 What do we all want to get out of this?

5.1.1 Yael

Learn about classical computation, learn language and formalism and then move past it. Then get into physical processes in terms of information theory and computation. Then move towards the cookbook. Look at continuous systems for computation rather than discrete.

5.1.2 Han

Physics, information, and complexity. Physics is understood,

Information is information theory and is related to physics, relate this to quantum computing.

Complexity, as in computational complexity, is an important research area.

5.1.3 Ben

Three main research areas: AFSR, come up with a new quantum algorithm(!), and theory of subsets of computation, possibly related to neural networks.

Interest for this class: Mirror of Neil's class: A few weeks of undergraduate physics, then transitioning to research. So, a few weeks of computer science, then transitioning to research areas.

Ben on complexity Phase space systems: computation is two things, reduction in phase space volume and parsing phase space into discrete units.

5.1.4 Ed

Work on problem from top-down and bottom up at same time. Work on small problems, how information propagates through interesting but tractable systems. Creation of a hierarchy of resources. What are the trade-offs; what do five extra electrons buy you? Nature seems to be doing its own computation: how does this work, and on a deep philosophical level deciding on what is really computation.

Launch both top-down and bottom up this summer. Categorize the computation which different things can do. Build a computation tool-kit of physical systems.

5.1.5 E'beth

Examine the open questions. Bullshit sessions in addition to formal classes — meet over dinner or something.

5.1.6 Rich

Fundamentally, what is computation? What can we say without making architecture-dependent assumptions? How does this relate to computational complexity/NP-complete? Can we create a language to describe computation that does not make architecture assumptions?

6 Questions for next time

6.1 Ed

Power of systems:

1. information flow
2. message passing, parallelism (bandwidth), "agent architectures," correlation.
3. Hopfield, Ising question $\rightarrow$ build tool-box for computation.

6.2 Yael

Black box computation: Putting signal into box, box performs operation. Can box be powered parasiticly? How much energy is required to do computation?

6.3 Ben

AFSR question. State vector v, matrix M, non-linear function g.

$$[g(M \cdot v)]$$

Understand shape of manifold based on $g = GF(2)$.

6.4 Rich

Three things:

1. Find universal definition of computing, phrased in terms of physical requirements (*e.g.* energy) but independent of any one physical system or computation architecture.
2. Find a language for describing universal computation, preferably at the level of writing useful programs.
3. Create a competent go playing system, preferably described in this language.

6.5 Han

Neural networks vs. neural computation — issues of filtering, encoding, and representation.

Filtering estimation and hypothesis testing.

Encoding preceptron model.

Question of how to relate filtering/encoding to representation. Relating this to neural structure.