Algorithms for Media

David Eppstein
Univ. of California, Irvine
School of Information and Computer Science

Joint work with Jean-Claude Falmagne
Univ. of California, Irvine
School of Social Sciences

What is a medium?

Introduced by Falmagne et al.
as a model of political choice theory

Encompasses many familiar combinatorial objects:

topological orderings of DAGs

acyclic orientations of undirected graphs

binary trees

cells in a hyperplane arrangement

antimatroids...

What is a medium?

Set of ***states*** S_0, S_1, S_2, ... acted on by a set of ***tokens*** t_0, t_1, t_2, ...
satisfying the following axioms:

1. Each token has a unique ***reverse***
If t takes state S_i to different state S_j, then ~t takes S_j to S_i
Reverse of reverse is original token

2. Any two states can reach each other by some sequence of tokens that does not contain both a token and the reverse of the same token

3. If a sequence of tokens takes a state to itself, and each token in the sequence changes the state, then for each token t the sequence contains equal numbers of t and of its reverse

4. If two sequences of tokens both take S_i to S_j, neither sequence contains a token and its reverse, and each step in each sequence changes the state, then both sequences are permutations of each other.

Media are like...

Media can be viewed as, and inherit properties of,
several more general types of object:

Deterministic finite state machines

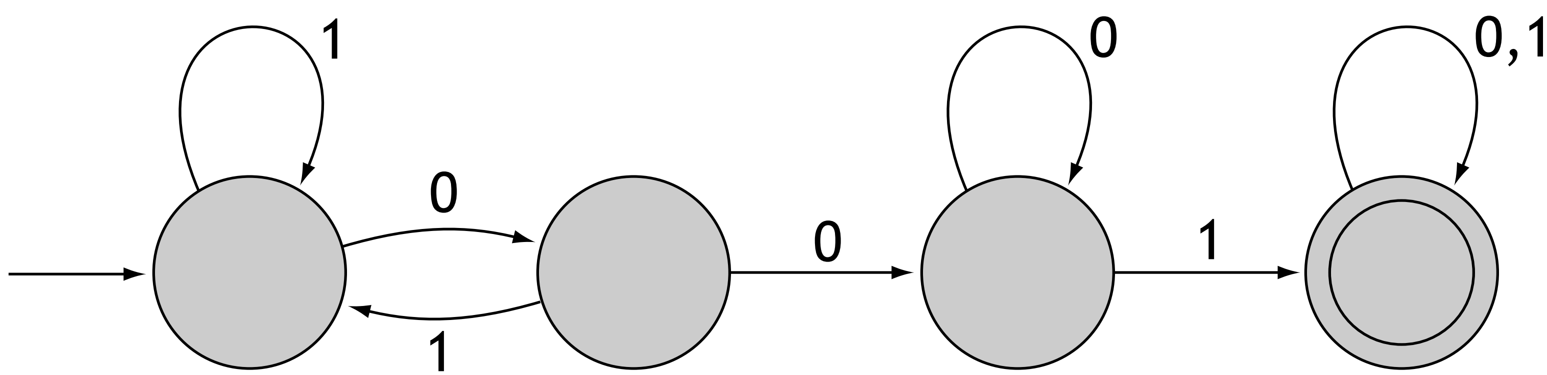
Undirected graphs

Points on a hypercube

Media as finite state machines

Finite state machine = set of states acted on by tokens

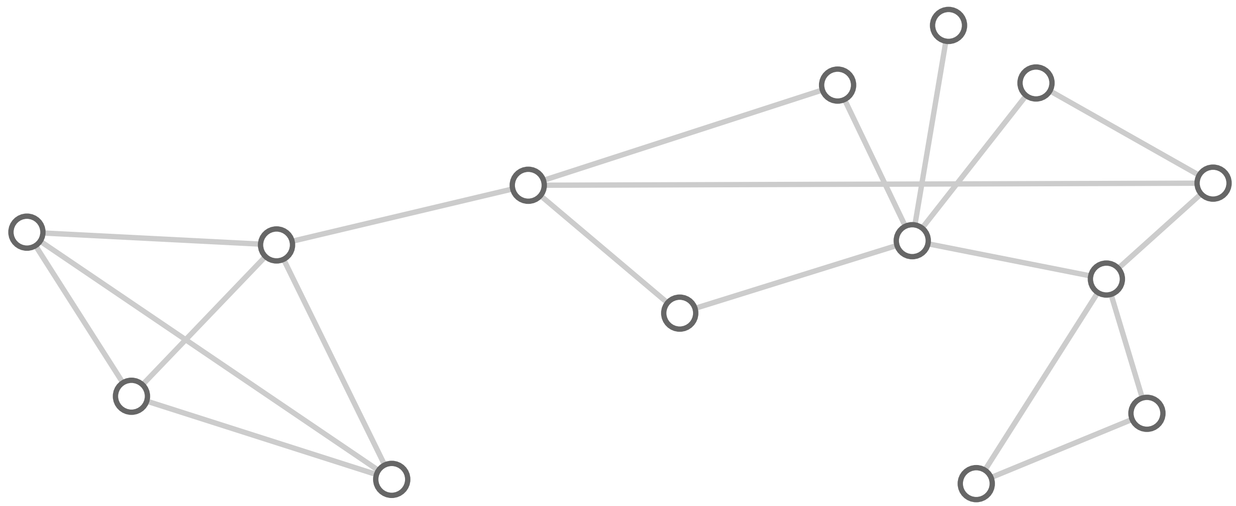
Used for:
Lexical analysis in programming language compilation
Controller design in integrated circuits
Pattern matching in text processing applications



Need not satisfy medium axioms
Typically also includes distinguished start state and accepting states

Media as undirected graphs

Undirected graph = set of vertices connected in pairs by edges



Graph of a medium:

One vertex per state

Edge from Si to Sj if some token takes Si to Sj

Undirected nature of the graph

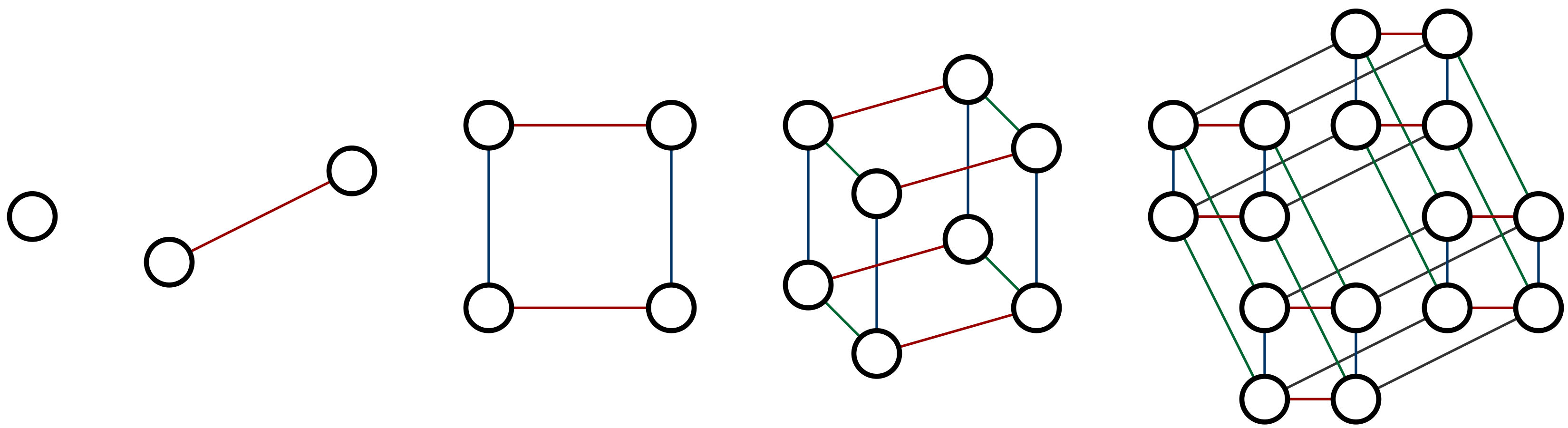
models the existence of reverse tokens in the medium

(can transition from S_i to S_j iff can go from S_j to S_i)

Media as hypercubes

Hypercube = Cartesian product of unit intervals
= convex hull of points with all coordinates zero or one

Hamming distance (L_1 metric) between vertices
= number of coordinates at which they differ



Every medium embeds isometrically into a hypercube
(coordinate = token-reverse pair)

Goals

1. Represent media as computer data

Allow typical basic operations to be done efficiently
Avoid using too much storage space

2. Explore algorithmic problems on media

What sorts of things do we want to compute?
Convert between representations
Work by analogy from DFAs, graphs, geometry

3. Find efficient algorithms

Analyze and compare worst-case asymptotic complexity
Improve on methods that don't use special structure of media

Measures of problem size

Want to describe storage space or algorithm runtime as function of input size
So, what is the size of a medium?

Multiple possible parameters:

n — number of states in the medium
τ — number of token pairs in the medium
m — number of adjacent pairs of states in the medium

To analyze algorithms, need to be able to compare parameters

$$\tau \le n \le m \le n \log_2 n \le n\,\tau$$

Representations of Media

Adjacency matrix

represent states as indices into state table
for each pair (state, token), store result of applying token to state
fast lookup of transitions: $O(1)$
somewhat high space: $O(n\ \tau)$

Adjacency list

each state can be represented in a single memory cell
for each state, store set of adjacent states and tokens leading to them
slower lookup of transitions: $O(\log \tau)$ or $O(1)$ with hashing
space-efficient: $O(m)$

Black box

each state can be stored in a known amount s of memory
input = single state and transition function (as callable subroutine)
very low space (only enough to store a single state)
appropriate for very large media (too large to store explicitly)

New algorithms

All pairs shortest paths

Find matrix of first steps on paths between each pair of states
Time = $O(n^2)$, improves $O(nm)$ time using graph BFS

Complementary pairs

Find pairs of states τ transitions apart (if any exist)

Time = $O(n\tau)$, improves $O(n^2)$ APSP, $O(n\tau \log n)$ hypercube embedding

Closed orientations

Test if given orientation is closed $O(m \log n)$; find closed orientation $O(m\tau)$

Reset sequences

Sequence of at most $n-1$ tokens takes all states to a single common state
Improves $O(n^3)$ length bound for arbitrary finite automata

Black box media

List all states, time = $O(n\tau^2)$, space $O(s)$
Based on Avis-Fukuda reverse search technique

Improves $O(ns)$ space, slightly slower than $O(n\tau \log n)$ time for DFS

New algorithms

All pairs shortest paths

Find matrix of first steps on paths between each pair of states
Time = $O(n^2)$, improves $O(nm)$ time using graph BFS

Complementary pairs

Find pairs of states τ transitions apart (if any exist)

Time = $O(n\tau)$, improves $O(n^2)$ APSP, $O(n\tau \log n)$ hypercube embedding

Closed orientations

Test if given orientation is closed $O(m \log n)$; find closed orientation $O(m\tau)$

Reset sequences

Sequence of at most n–1 tokens takes all states to a single common state
Improves $O(n^3)$ length bound for arbitrary finite automata

Black box media

List all states, time = $O(n\tau^2)$, space $O(s)$
Based on Avis-Fukuda reverse search technique

Improves $O(ns)$ space, slightly slower than $O(n\tau \log n)$ time for DFS

All pairs shortest paths algorithm: main ideas

Visit all states in depth first order

While visiting each state Q, maintain:

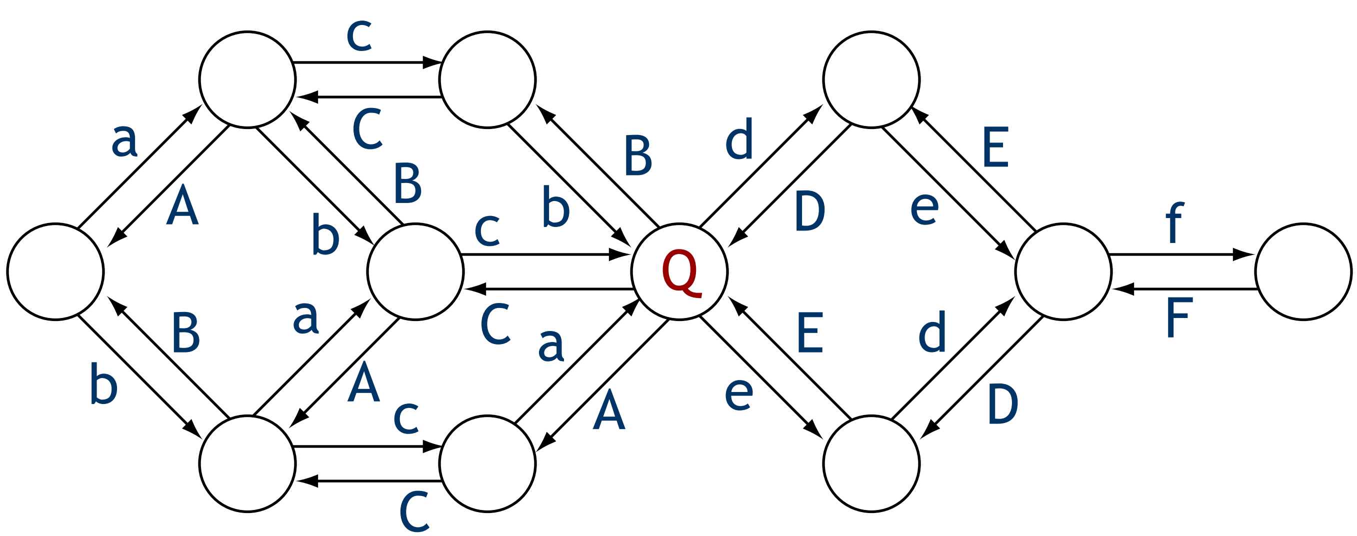
Ordered list L of tokens that can appear on shortest paths to Q
(one list item for each token-reverse pair)

Pointer from each other state to the first effective token t in L

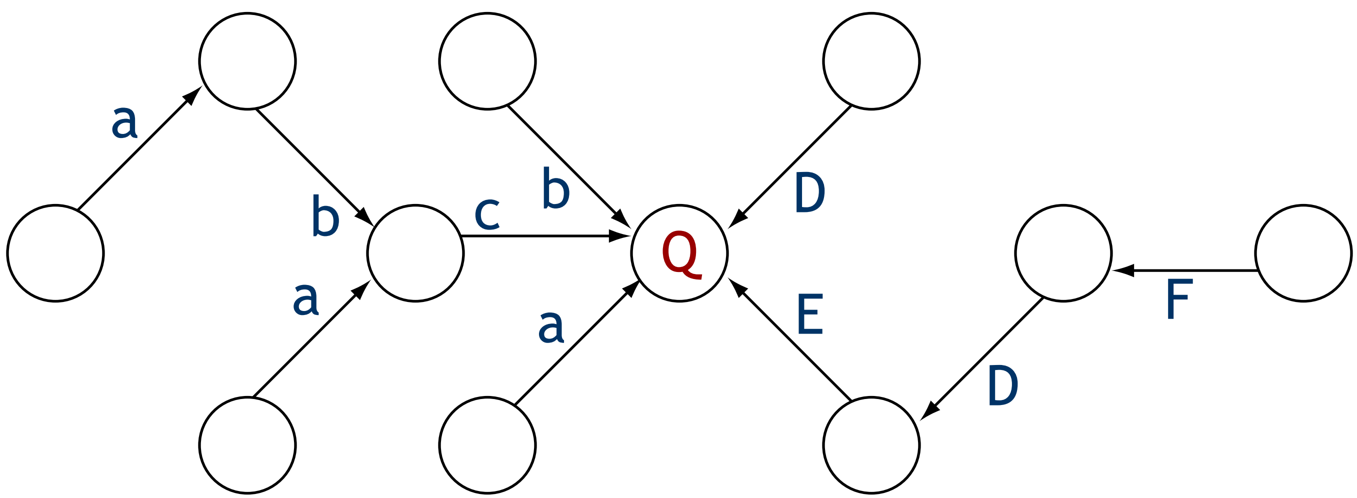
The effective transitions from this set of pointers
form a tree of shortest paths to Q

By visiting all states, we find trees of shortest paths to all states in the medium

All pairs shortest paths: example



L: a, b, c, D, E, F



First transitions in L from each state

All pairs shortest paths algorithm: single step of traversal

To move from state Q to Q':

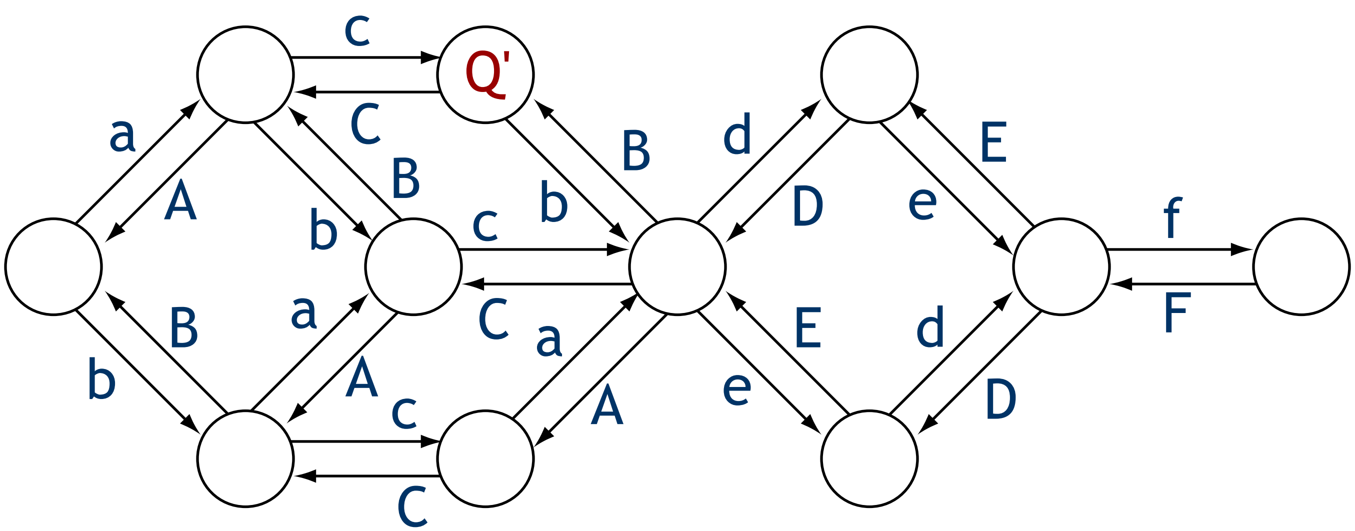
Find token t taking Q' to Q

Remove t from ordered list L

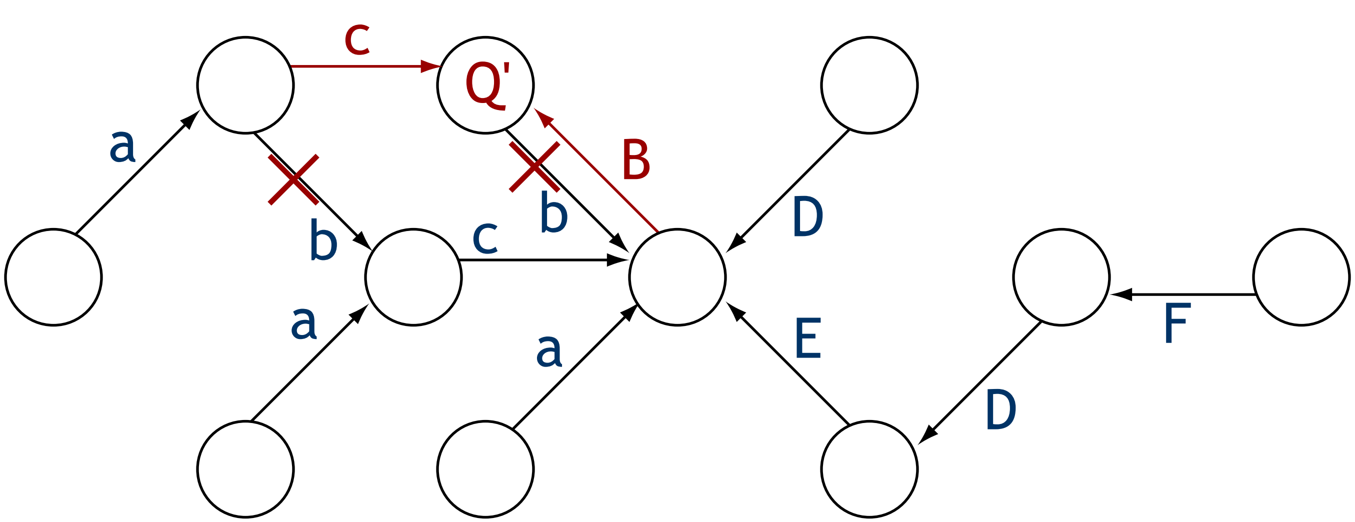
Add reverse(t) to end of L

For each state S for which t was first effective token in L:
search sequentially along L for next effective token

All pairs shortest paths: example revisited



L: a, ~~b~~, c, D, E, F, B



First transitions in L from each state

All pairs shortest paths algorithm: analysis

DFS performs O(n) transitions
Each transition adds one more token to end of L
    Total number of additions to L: O(n)

Each of n states repeatedly searches L
Searches never revisit previously-searched parts of L
So total steps per state = total additions to L
    Total time spent performing sequential searches: $O(n^2)$

Other steps of algorithm (outputting shortest path trees)
can be done in time O(n) per visited state
    Total time for algorithm: $O(n^2)$

Conclusions

Efficient computer representations of media

Efficient algorithms for several fundamental problems

Algorithms are simple and implementable
(black box traversal is already implemented)

Directions for future work

Additional efficient media-theoretic algorithms
e.g. o(mn) recognition of graphs of media?

Identify applications where efficient implementation is important