Homing and Synchronizing Sequences

Sven Sandberg

Information Technology Department
Uppsala University
Sweden

Outline

1. Motivations
2. Definitions and Examples
3. Algorithms
   (a) Current State Uncertainty (used in algorithms)
   (b) Computing Homing Sequences
   (c) Computing Synchronizing Sequences
4. Variations
   (a) Adaptive homing sequences
   (b) Computing shortest sequences
   (c) Parallel algorithms
   (d) Difficult related problems
5. Conclusions

Motivation for Homing Sequences: Testing [13]

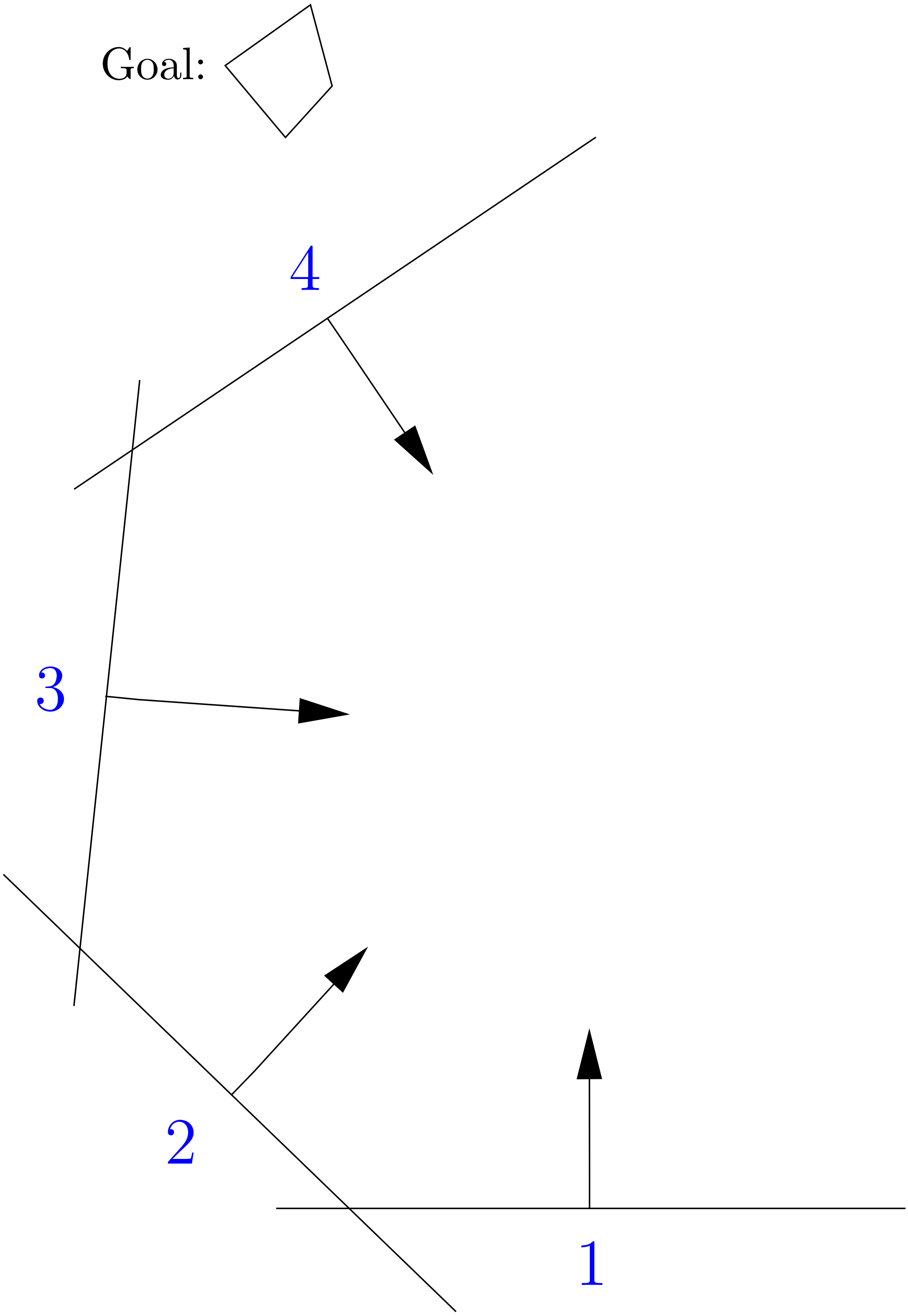
- Learning algorithms:

  experiment with a given a black box automaton
  until you learn the contents

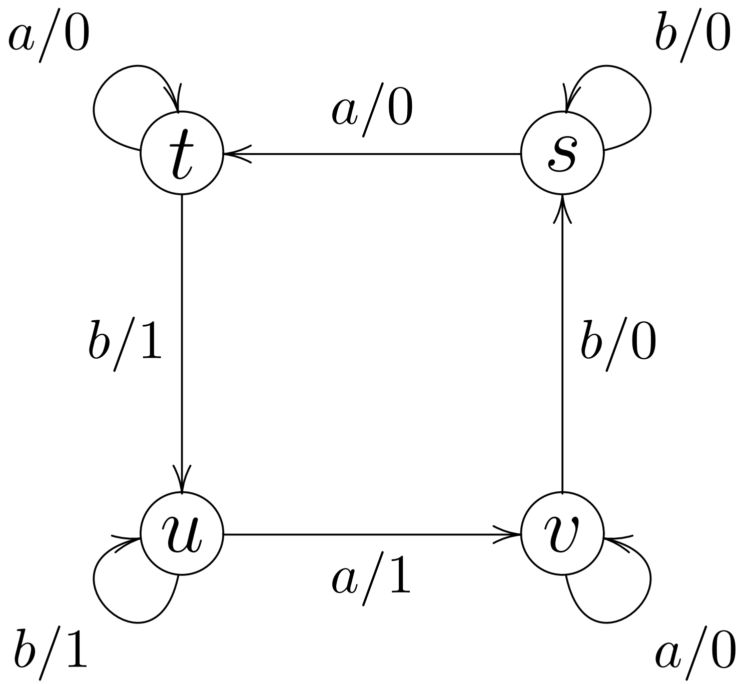
- Protocol verification

- Hardware fault-detection

Motivation for Synchronizing Sequences: Pushing Things [12]



Mealy Machines



Deterministic, *total*, finite state machine with outputs on transitions

Inputs: $I = \{a, b\}$

Outputs: $O = \{0, 1\}$

States: $S = \{s, t, u, v\}$

Mealy machine: $\mathcal{M} = \langle I, O, S, \delta, \lambda \rangle$

Inputs, I

Outputs, O

States, S

transition function ("arrows"), $\delta : S \times I \to S$

output function, $\lambda : S \times I \to O$

Synchronizing Sequences [12]

Intuitive Definition

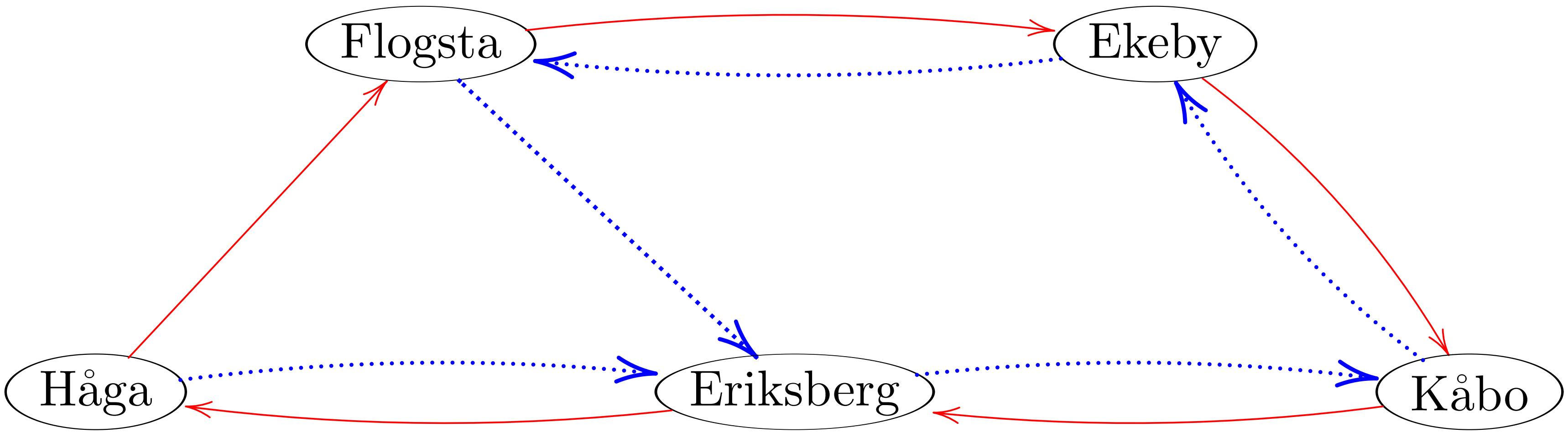
1. Initial state is unknown.
2. Apply a sequence $x \in I^*$ of inputs,
3. afterwards only one final state is possible

If this is possible, x is a **synchronizing sequence**

Formal Definition

$x \in I^*$ is synchronizing iff $|\delta(S, x)| = 1$

Example: Getting Home by Subway in Uppsala [14]



- Initial position is unknown
- There are no signs that reveal the current station
- Find your way to Flogsta, switching red and blue line as needed

Solution: brrbrrbrrbrr

Homing Sequences

Intuitive Definition

1. Initial state is unknown.
2. Apply a sequence $x \in I^*$ of inputs,
3. observe outputs,
4. conclude what the *final* state is

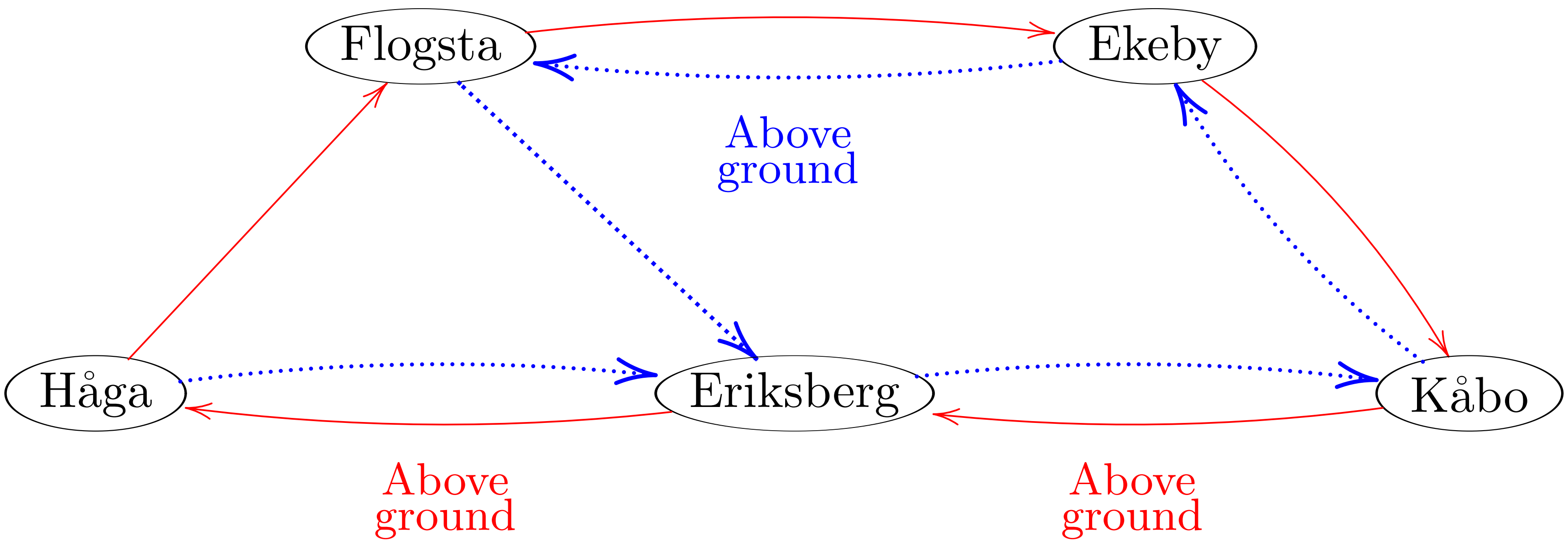
If this is possible, x is a **homing sequence**

Formal Definition

$x \in I^*$ is homing iff
for all states $s, t \in S$, $\quad \delta(s,x) \neq \delta(t,x) \implies \lambda(s,x) \neq \lambda(t,x)$

Homing Sequences: Example

- Homing sequences care about the output
- E.g., in Uppsala the subway sometimes goes above ground.
- Using this information, we can more efficiently figure out the final state.

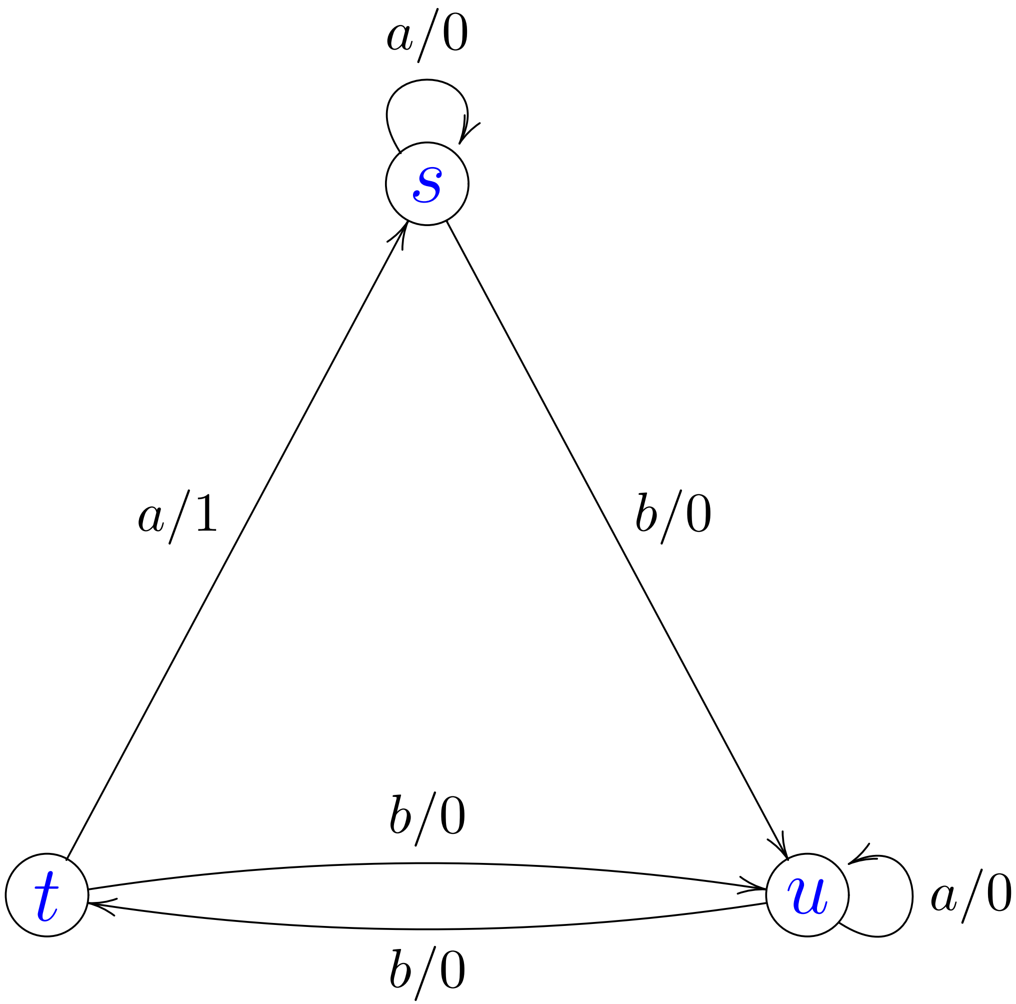


Solution: e.g., brr

Initial State Uncertainty [14]

- Data structure crucial in algorithms computing homing sequences
- The Initial State Uncertainty with respect to an input string
  "indicates for each output string the set of possible initial states"
- Formally, for an input string $x \in I^*$ it is the partition of states
  induced by the equivalence relation
  $s \equiv t \iff \lambda(s, x) = \lambda(t, x)$
  ("x produces the same output from s as from t")

Initial State Uncertainty: Example



Initial State Uncertainty: Example

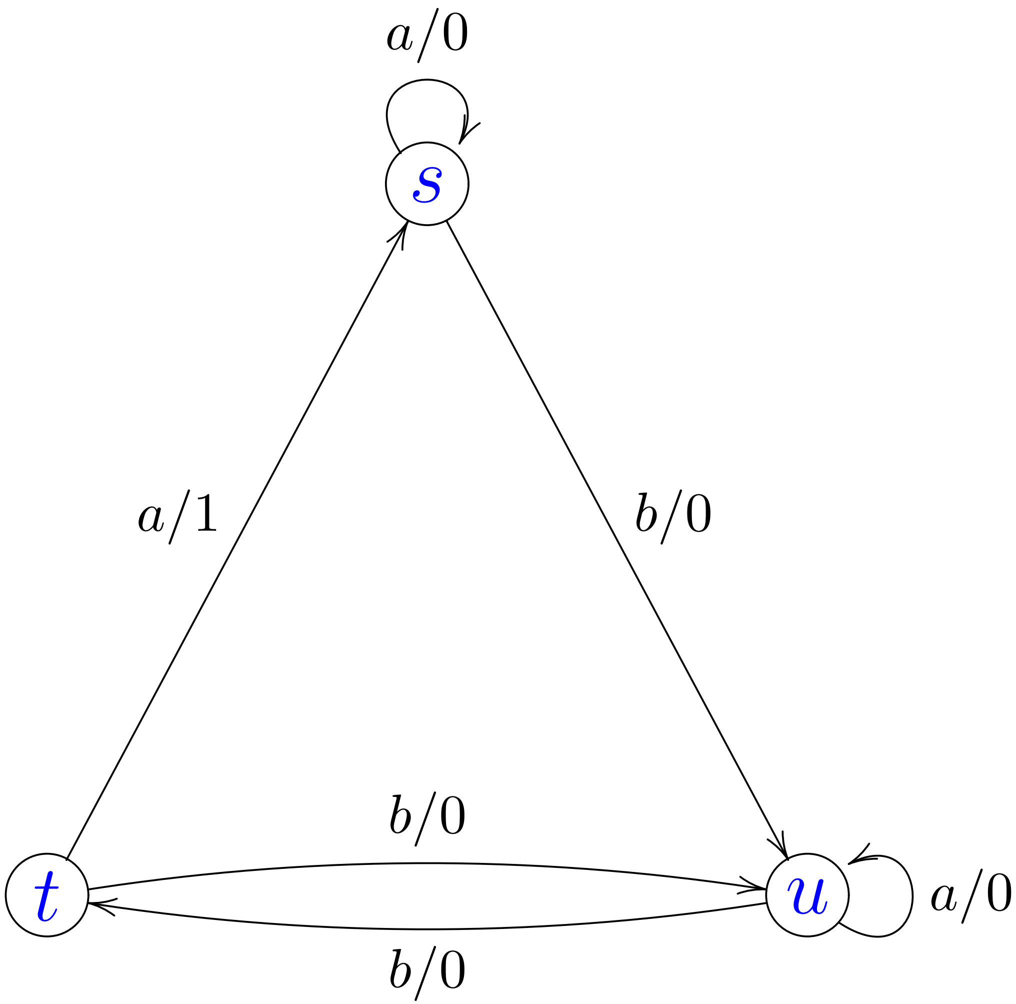
input string	initial state uncertainty
ε	$\{\{s, t, u\}\}$
a	$\{\{t\}_1, \{s, u\}_0\}$
ab	$\{\{t\}_{10}, \{s, u\}_{00}\}$
aba	$\{\{t\}_{100}, \{s\}_{000}, \{u\}_{001}\}$

(here, the output corresponding to a block is indicated in red)

Current State Uncertainty [15]

- Another data structure crucial in algorithms computing homing sequences
- The Current State Uncertainty with respect to an input string "indicates for each output string the set of possible final states"
- Formally, for an input string $x \in I^*$ it is the set $\sigma(x) \stackrel{\text{def}}{=} \{\delta(B, x) : B \text{ is a block of the initial state uncertainty w.r.t. } x\}$.
- **Important:** x is homing iff $\sigma(x)$ is a set of singletons

Current State Uncertainty: Example



Current State Uncertainty: Example

input string	initial state uncertainty	current state uncertainty
ε	$\{\{s,t,u\}\}$	$\{\{s,t,u\}\}$
a	$\{\{t\}_1, \{s,u\}_0\}$	$\{\{s\}_1, \{s,u\}_0\}$
ab	$\{\{t\}_{10}, \{s,u\}_{00}\}$	$\{\{u\}_{10}, \{u,t\}_{00}\}$
aba	$\{\{t\}_{100}, \{s\}_{000}, \{u\}_{001}\}$	$\{\{u\}_{100 \text{ or } 000}, \{s\}_{001}\}$

Computing Homing Sequences: Idea

Assume machine is minimized.

- Concatenate strings iteratively,
- in each step improving the current state uncertainty.
  ("$\sum_{B \in \sigma(x)} |B| - |\sigma(x)|$" decreases)
- Each string should be *separating* for two states in the same block:

  A *separating sequence* $x \in I^*$ for two states $s, t \in S$ gives different outputs: $\lambda(s, x) \neq \lambda(t, x)$

  Since the machine is minimized, separating sequences always exist

Computing Homing Sequences: Algorithm

function HOMING-FOR-MINIMIZED(Minimized Mealy machine $\mathcal{M}$)
    $x \leftarrow \varepsilon$
    while there is a block $X \in \sigma(x)$ with $|X| > 1$
        take two different states $s, t \in X$
        let y be a separating sequence for s and t
        $x \leftarrow xy$
    return x

Homing Sequences: Quality of Algorithm

(n = number of states, $|I|$ = number of input symbols)

- Time: $O(n^3 + n^2 \cdot |I|)$
- Space: $O(n)$
  (not counting the space needed by the output)
- Sequence length: $\leq n(n-1)/2$
  Some machines require $\geq n(n-1)/2$

Computing Synchronizing Sequences: Idea **[17]**

Very similar to algorithm for homing sequences:

- Concatenate strings iteratively,
- in each step decrease $|\delta(S, x)|$.
- Each string should be *merging* for two states in $\delta(S, x)$:
  - A *merging sequence* $y \in I^*$ for two states $s, t \in S$ takes them to the same final state: $\delta(s, y) = \delta(t, y)$
  - This guarantees that $|\delta(S, xy)| < |\delta(S, x)|$
  - Merging sequences exist for all states $\iff$ there is a synchronizing sequence

Computing Synchronizing Sequences: Algorithm

[18]

Very similar to algorithm for homing sequences:

function SYNCHRONIZING(Mealy machine $\mathcal{M}$)
    $x \leftarrow \varepsilon$
    while $|\delta(S, x)| > 1$
        take two different states $s, t \in \delta(S, x)$
        let y be a merging sequence for s and t
          (if none exists, **return** FAILURE)
        $x \leftarrow xy$
    return x

Synchronizing Sequences: Quality of Algorithm

[19–20]

- Time: $O(n^3 + n^2 \cdot |I|)$
- Space: $O(n^2 + n \cdot |I|)$
  (not counting the space needed by the output)
- Sequence length: $\leq (n^3 - n)/6$

  Černý's conjecture: length $\leq (n-1)^2$
  (true in special cases, open in general)

  Some machines require length $\geq (n-1)^2$

Homing Sequences for General Machines [20–21]

- We don't need to assume the machine is minimized
- A different algorithm solves this more general problem, but less efficiently

  Combines ideas from algorithms for homing and synchronizing sequences
- Often possible to assume the machine is minimized

Adaptive Homing Sequences [21–22]

- Apply the sequence as it is being computed,
- and let current input depend on previous outputs

- Can use modified version of the usual homing sequence algorithm

- May result in shorter sequence,
- but equally long in the worst case: $(n-1)^2$

Finding the Shortest Sequence [24–26]

- It is important to minimize the length of sequences:
  - recall pushing things
  - in testing, a machine may be remote or very slow
- Exponential algorithms have been used
- Unfortunately, the problems are NP-complete
- Even impossible to approximate unless P=NP
  (follows from NP-completeness proof)

Related Problems are PSPACE-complete [26–28]

1. Nondeterministic transition system
   (instead of deterministic)
2. The initial state is in a subset $X \subseteq S$
   (instead of S)
3. The final state may be in a subset $X \subseteq S$
   (instead of any single state in S)

Parallel Algorithms [29]

Homing Sequences

- Randomized algorithm uses $\log^2 n$ time, $O(n^7)$ processors.
  Hence, the problem belongs to RNC.
- Deterministic algorithm uses $O(\sqrt{n}\log^2 n)$ time
  Impractical due to high communication cost
- There is also a practical randomized algorithm

Synchronizing Sequences

- No known parallel algorithm
- Except one for *monotonic automata*

Conclusion

Homing sequences

- Problem is more or less solved (optimal and polynomial algorithm is known)
- Apparently more used for testing than synchronizing sequences

Synchronizing sequences

- Open question:
  Narrow the gap between upper bound $O(n^3)$ and lower bound $\Omega(n^2)$
  for the length of sequences
- Interesting algebraic properties and other applications,
  but less used for testing