

Logical and Stochastic Modeling

Gianfranco Ciardo

Copyright ©2004 Gianfranco Ciardo
All rights reserved

EXPLICIT STATE-SPACE GENERATION

We consider a discrete-state model that defines:

- a potential state space $\hat{\mathcal{X}}$ the “type” of the state
- the set of initial states $\mathcal{X}_{init} \subseteq \hat{\mathcal{X}}$ often there is a single initial state \mathbf{x}_{init}
- a set of events \mathcal{E} defining a disjunctively-partitioned next-state function
 - $\mathcal{N}_\alpha : \hat{\mathcal{X}} \rightarrow 2^{\hat{\mathcal{X}}}$ $\mathbf{j} \in \mathcal{N}_\alpha(\mathbf{i})$ iff state \mathbf{j} can be reached by firing event α in state \mathbf{i}
 - $\mathcal{N} : \hat{\mathcal{X}} \rightarrow 2^{\hat{\mathcal{X}}}$ $\mathcal{N}(\mathbf{i}) = \bigcup_{\alpha \in \mathcal{E}} \mathcal{N}_\alpha(\mathbf{i})$
 - naturally extended to sets of states $\mathcal{N}_\alpha(\mathcal{X}) = \bigcup_{\mathbf{i} \in \mathcal{X}} \mathcal{N}_\alpha(\mathbf{i})$ and $\mathcal{N}(\mathcal{X}) = \bigcup_{\mathbf{i} \in \mathcal{X}} \mathcal{N}(\mathbf{i})$
 - α is enabled in \mathbf{i} iff $\mathcal{N}_\alpha(\mathbf{i}) \neq \emptyset$, otherwise it is disabled
 - \mathbf{i} is absorbing, or a trap, or dead iff $\mathcal{N}(\mathbf{i}) = \emptyset$

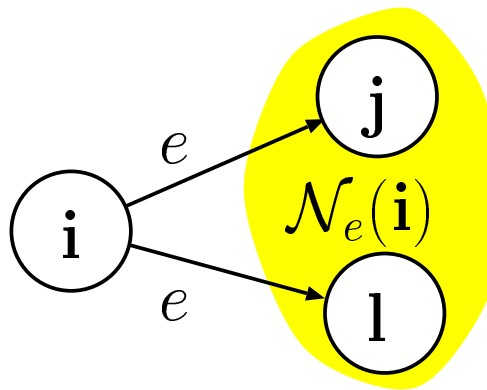
We often define a set \mathcal{E} of model **events** and decompose \mathcal{N} accordingly: $\mathcal{N}(\mathbf{i}) = \bigcup_{e \in \mathcal{E}} \mathcal{N}_e(\mathbf{i})$

Event e is **disabled** in state \mathbf{i} if $\mathcal{N}_e(\mathbf{i}) = \emptyset$

Event e is **enabled** in state \mathbf{i} if $\mathcal{N}_e(\mathbf{i}) \neq \emptyset$

If \mathbf{j} can be reached from \mathbf{i} when e **fires** (occurs), we write $\mathbf{i} \xrightarrow{e} \mathbf{j}$ or $\mathbf{j} \in \mathcal{N}_e(\mathbf{i})$

In a **nondeterministic** formalism, $\mathcal{N}_e(\mathbf{i})$ can contain multiple states



In a **deterministic** formalism, $\mathcal{N}_e(\mathbf{i})$ is either empty or it contains one state

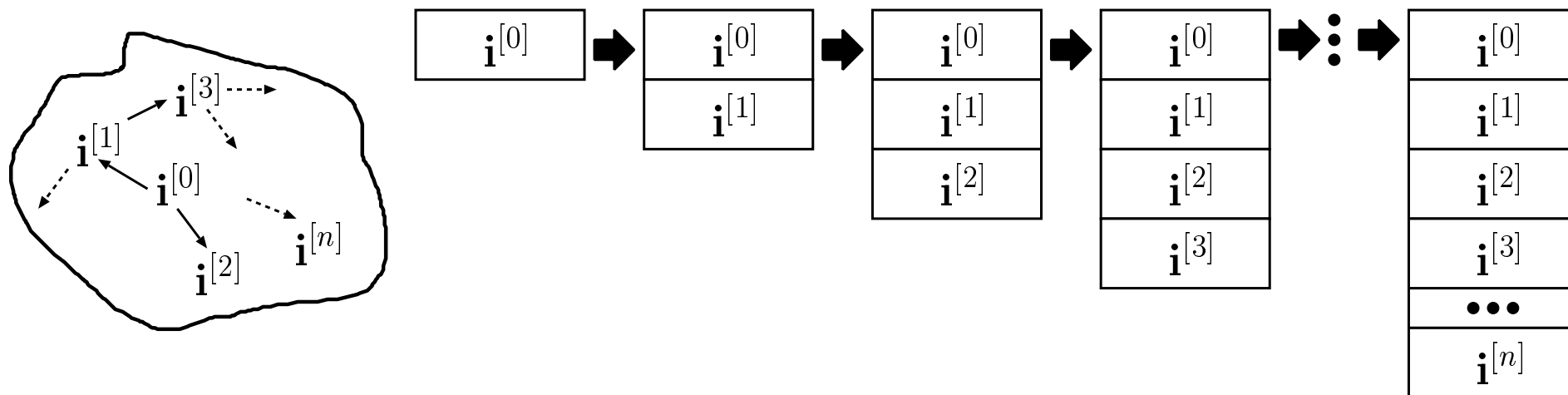
But nondeterminism still arises if multiple events are enabled in a state

The **state space** \mathcal{X}_{reach} of the model is the smallest subset of $\hat{\mathcal{X}}$ containing \mathcal{X}_{init} and satisfying:

- the **recursive definition** $\mathbf{i} \in \mathcal{X}_{reach} \wedge \mathbf{j} \in \mathcal{N}(\mathbf{i}) \Rightarrow \mathbf{j} \in \mathcal{X}_{reach}$
- or the **fixed-point equation** $\mathcal{Y} = \mathcal{Y} \cup \mathcal{N}(\mathcal{Y})$

$$\mathcal{X}_{reach} = \mathcal{X}_{init} \cup \mathcal{N}(\mathcal{X}_{init}) \cup \mathcal{N}^2(\mathcal{X}_{init}) \cup \mathcal{N}^3(\mathcal{X}_{init}) \cup \dots = \mathcal{N}^*(\mathcal{X}_{init})$$

Explicit generation of \mathcal{X}_{reach} adds **one state** at a time



with an explicit data structure

memory requirements increase monotonically during generation

they are proportional to $|\mathcal{X}_{reach}|$ at the end

State-by-state (explicit) generation algorithm

```

ExploreExplicit(  $\mathcal{X}_{init}$  : set of states,  $\mathcal{N}$  : next-state function ) : set of states
local  $\mathcal{U}, \mathcal{Y}$  : set of states;
local  $\mathbf{i}, \mathbf{j}$  : state;
1  $\mathcal{Y} \leftarrow \emptyset$ ;            $\mathcal{Y}$  contains the known states already explored
2  $\mathcal{U} \leftarrow \mathcal{X}_{init}$ ;      $\mathcal{U}$  contains the known states not yet explored
3 while  $\mathcal{U} \neq \emptyset$  do
4     choose a state  $\mathbf{i}$  in  $\mathcal{U}$  and move it from  $\mathcal{U}$  to  $\mathcal{Y}$ ;
5     for each  $\mathbf{j} \in \mathcal{N}(\mathbf{i})$  do
6         if  $\mathbf{j} \notin \mathcal{Y} \cup \mathcal{U}$  then           search to determine whether  $\mathbf{j}$  is a new state
7              $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathbf{j}\}$ ;       remember to explore  $\mathbf{j}$  later
8         end if;
9     end for;
10 end while;
11 return  $\mathcal{Y}$ ;

```

the memory requirements are $O(|\mathcal{X}_{reach}|)$

most time is spent searching for a state (line ??)

A structured discrete-state model is specified by

- a potential state space $\hat{\mathcal{X}} = \mathcal{X}_L \times \cdots \times \mathcal{X}_1$
 - the “type” of the (global) state
 - \mathcal{X}_k is the (discrete) local state space for the k^{th} submodel
 - if \mathcal{X}_k is finite, we can map it to $\{0, 1, \dots, n_k - 1\}$ n_k might be unknown a priori

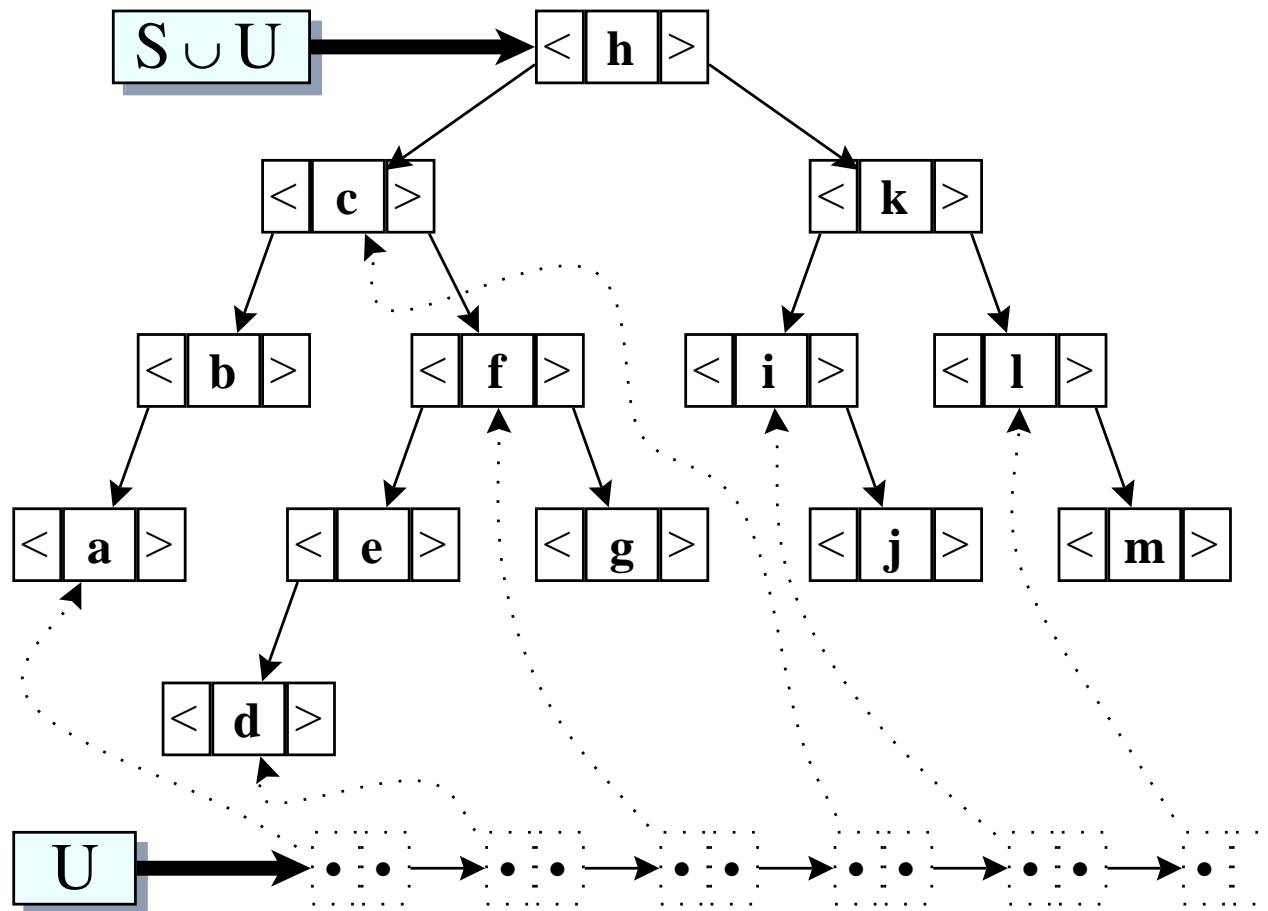
- a set of initial states $\mathcal{X}_{init} \subseteq \hat{\mathcal{X}}$
 - often there is a single initial state \mathbf{i}_{init}

- a set of events \mathcal{E} defining a disjunctively-partitioned next-state function or transition relation
 - $\mathcal{N}_\alpha : \hat{\mathcal{X}} \rightarrow 2^{\hat{\mathcal{X}}}$ $\mathbf{j} \in \mathcal{N}_\alpha(\mathbf{i})$ iff state \mathbf{j} can be reached by firing event α in state \mathbf{i}
 - $\mathcal{N} : \hat{\mathcal{X}} \rightarrow 2^{\hat{\mathcal{X}}}$ $\mathcal{N}(\mathbf{i}) = \bigcup_{\alpha \in \mathcal{E}} \mathcal{N}_\alpha(\mathbf{i})$
 - naturally extended to sets of states $\mathcal{N}_\alpha(\mathcal{Y}) = \bigcup_{\mathbf{i} \in \mathcal{Y}} \mathcal{N}_\alpha(\mathbf{i})$ and $\mathcal{N}(\mathcal{Y}) = \bigcup_{\mathbf{i} \in \mathcal{Y}} \mathcal{N}(\mathbf{i})$
 - α is enabled in \mathbf{i} iff $\mathcal{N}_\alpha(\mathbf{i}) \neq \emptyset$, otherwise it is disabled
 - \mathbf{i} is absorbing, or dead, if $\mathcal{N}(\mathbf{i}) = \emptyset$

How can we store S and U efficiently?

If we store S and U together, we can distinguish them using a linked list for U

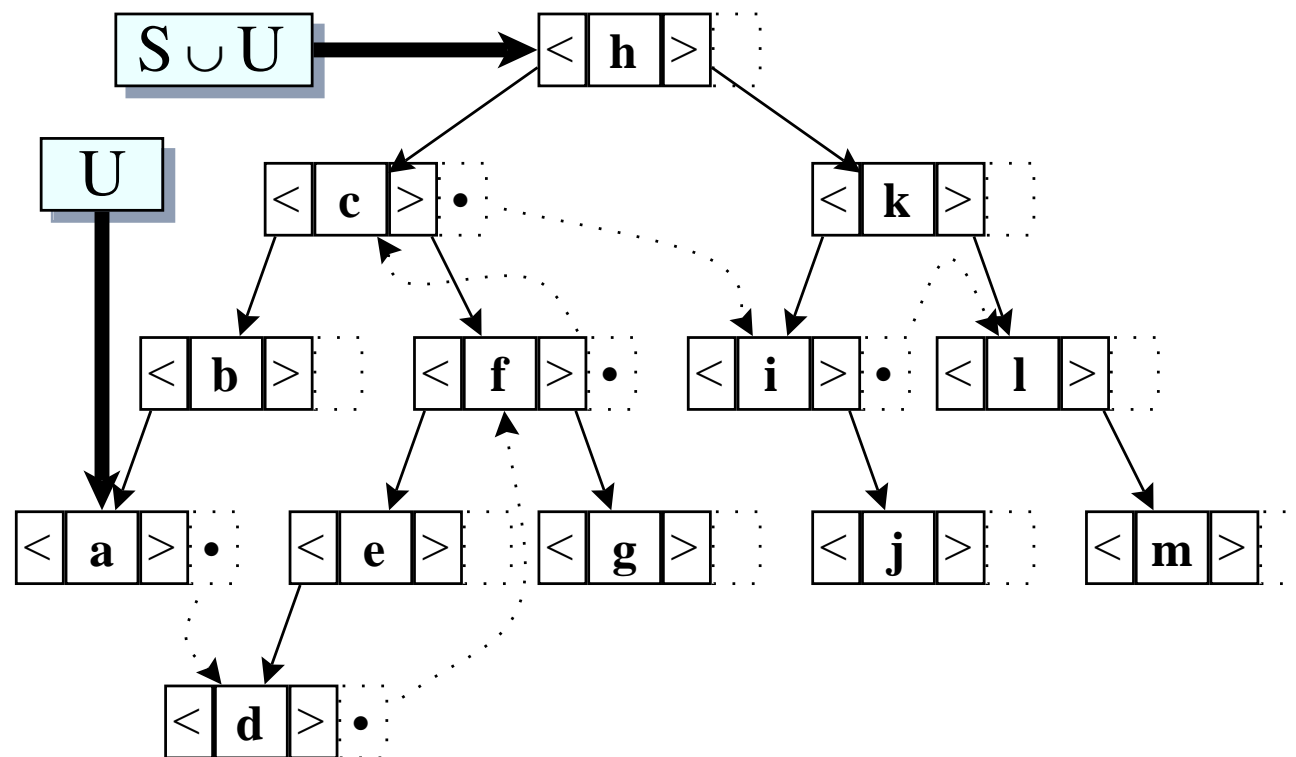
\Rightarrow Additional $2 \cdot |U| \cdot B_{pointer}$ bits



How can we store S and U efficiently? (cont.)

Or a pointer the next unexplored state, in each tree node

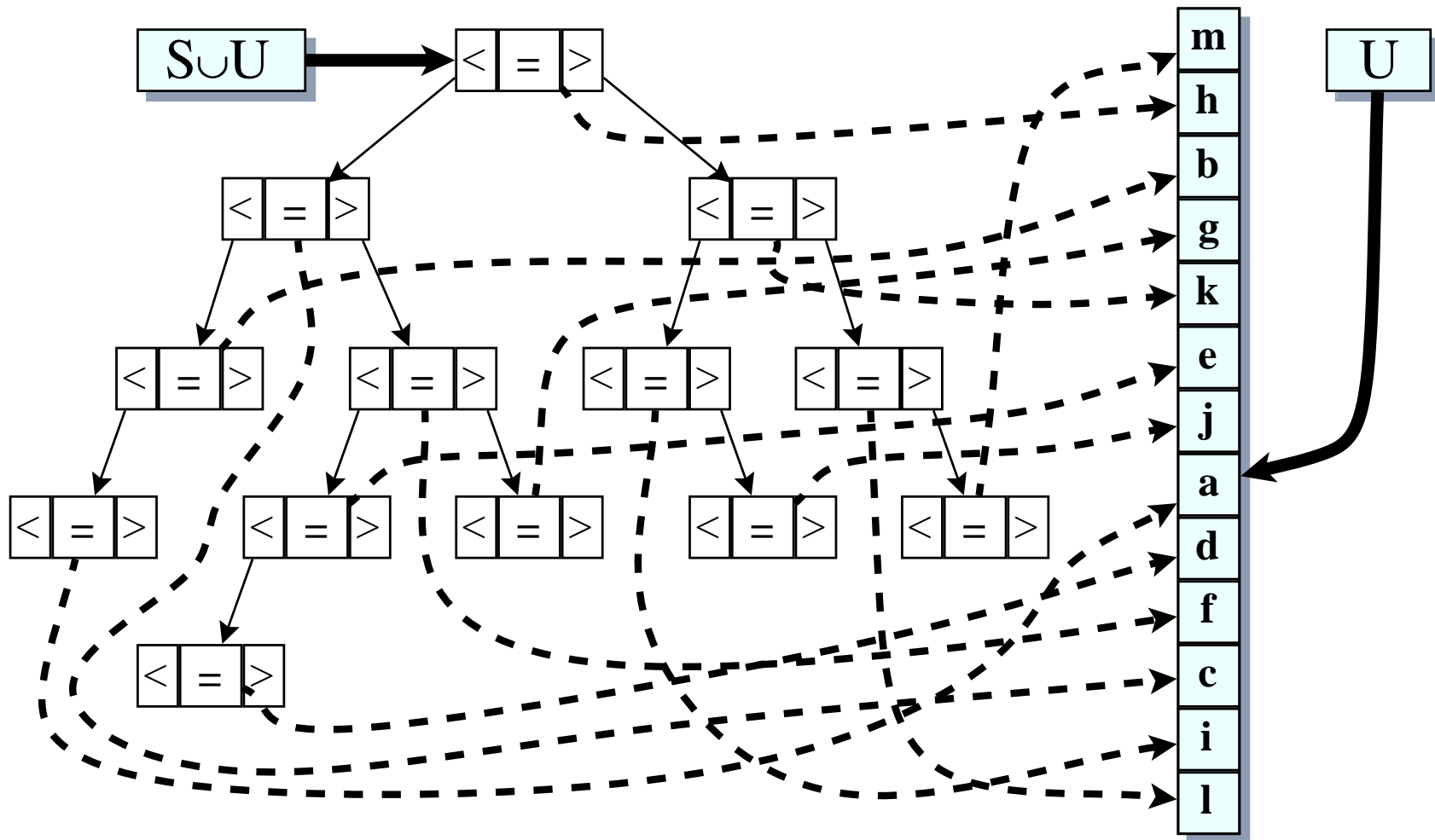
\Rightarrow Additional $|\mathcal{S}| \cdot B_{pointer}$ bits



How can we store S and U efficiently? (cont.)

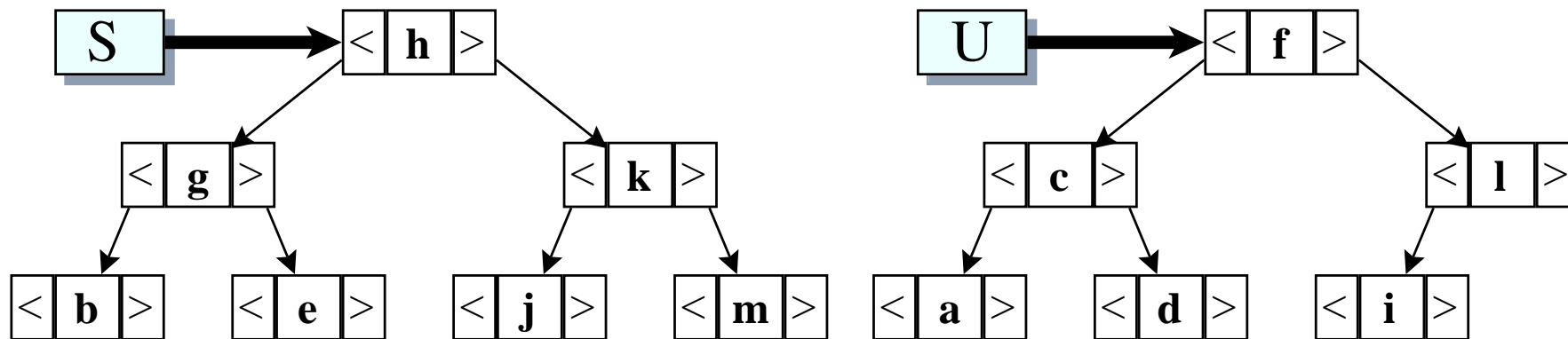
Or store the states in a dynamic array structure

⇒ Additional $|\mathcal{S}| \cdot B_{index}$ bits

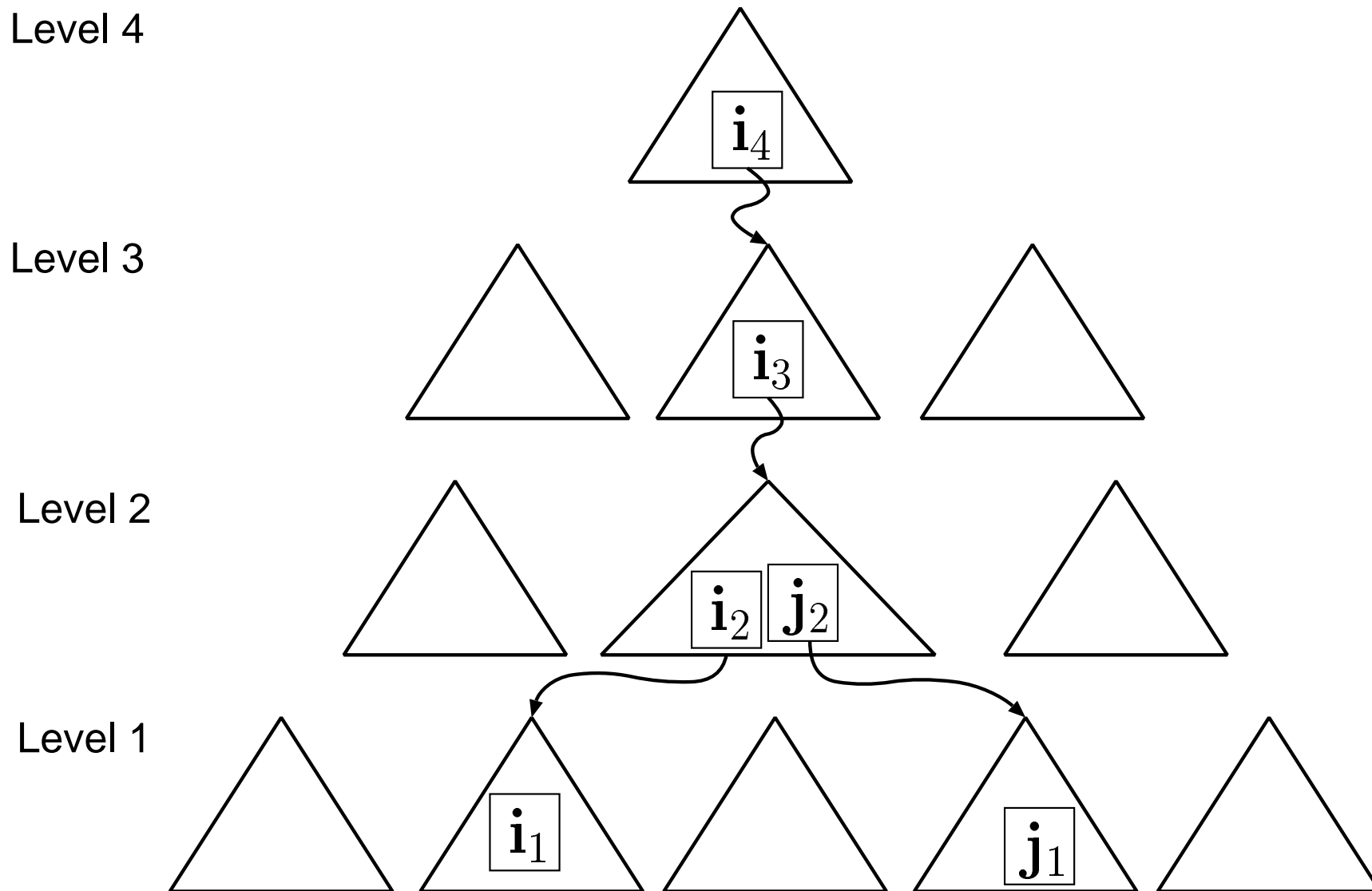


How can we store S and U efficiently? (cont.)

If we store S and U separately:



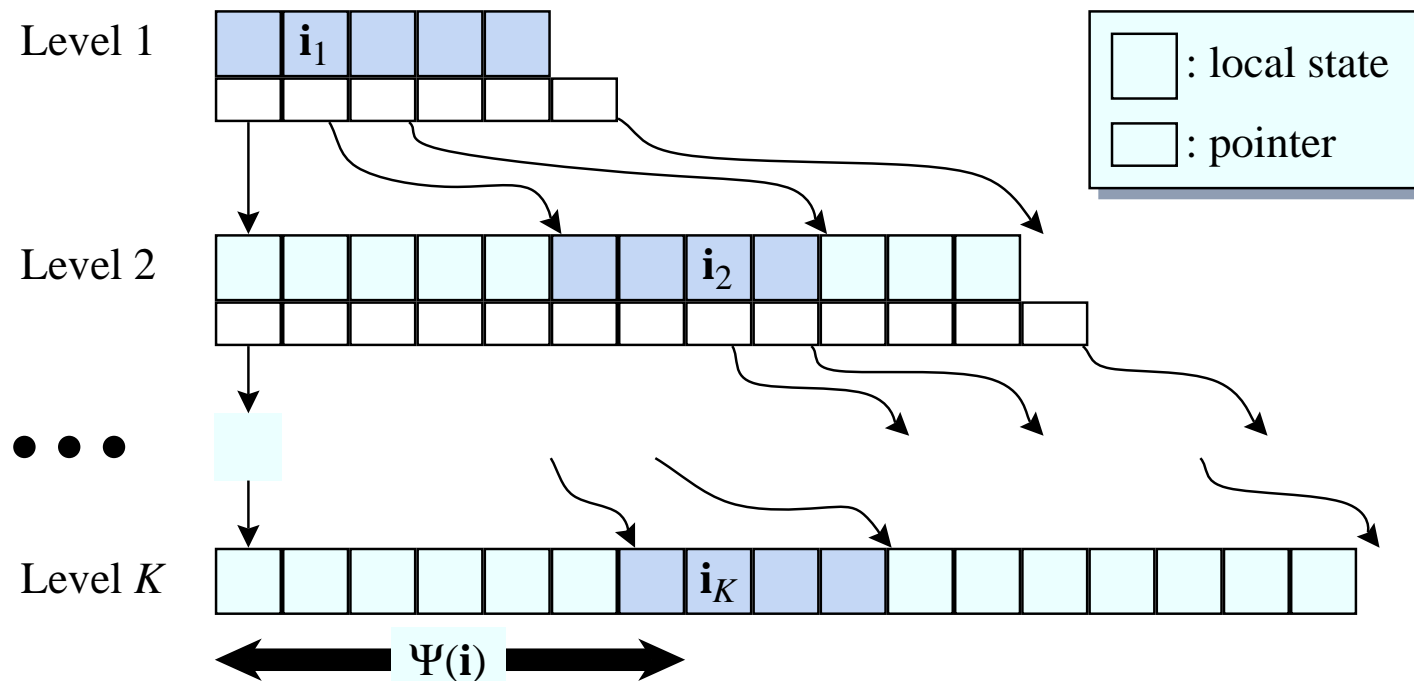
A multilevel data structure to store \mathcal{S}



memory requirements: little over $3 \lceil \log n_1 \rceil$ bits per state
(assuming good fanout from one level to the next)

Compressing \mathcal{S} after generation

Once \mathcal{S} has been built, we can compress it using arrays:

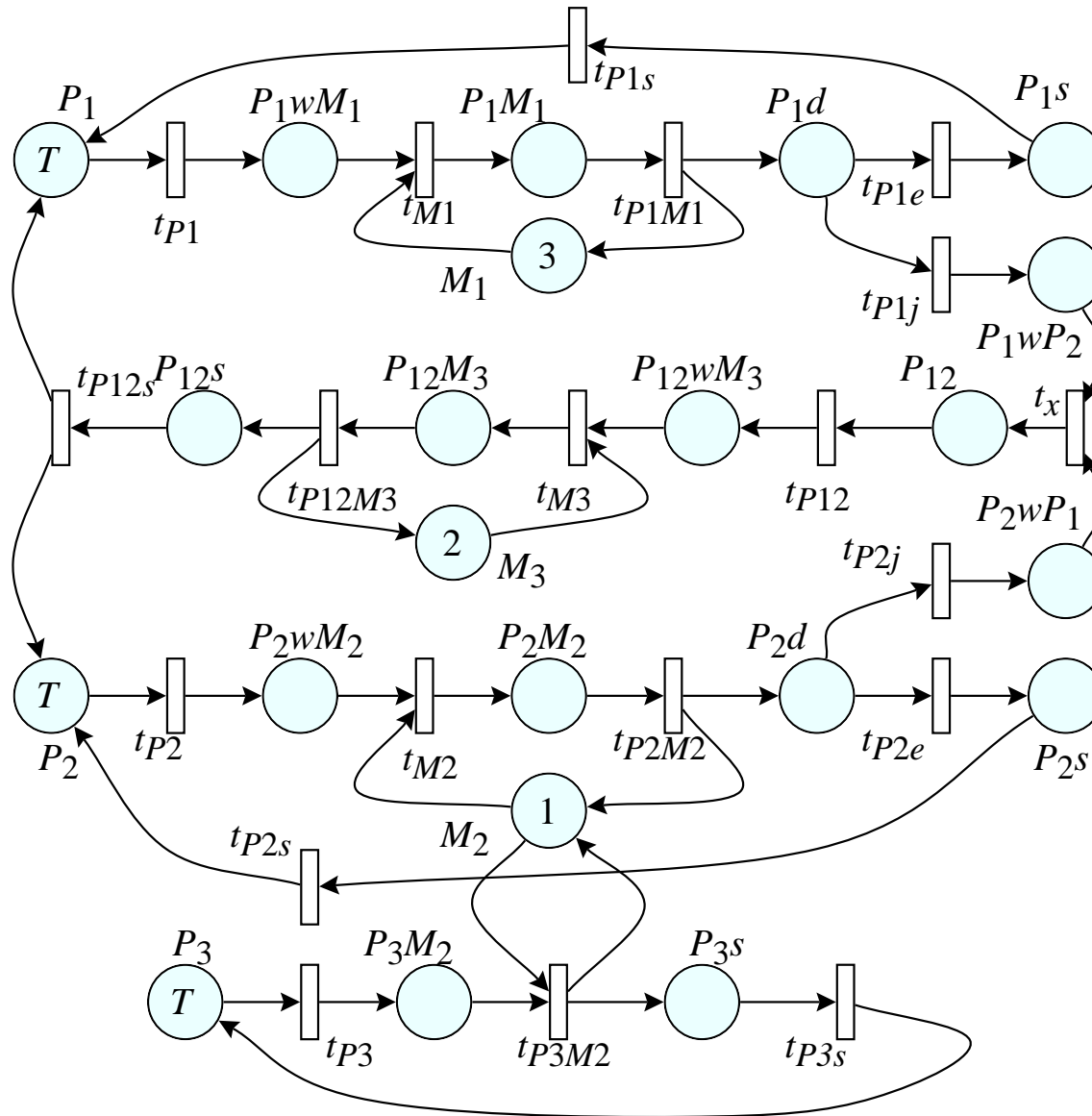


the distance $\psi(\mathbf{i})$ is the lexicographic index of \mathbf{i} in \mathcal{S}

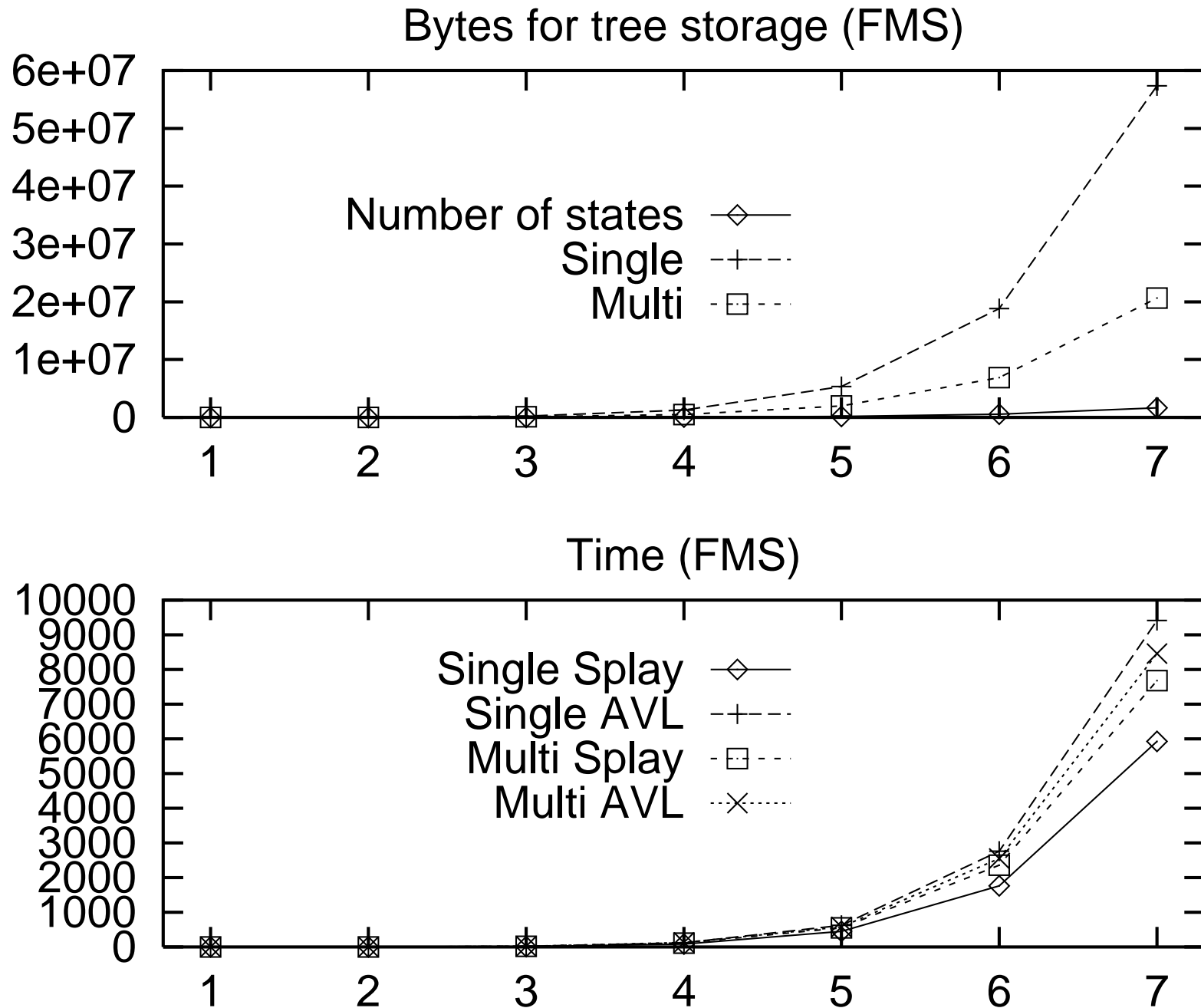
memory requirements: little over $\lceil \log n_1 \rceil$ bits per state
(assuming good fanout from one level to the next)

worst non-amortized search time: $\sum_{K \geq k \geq 1} \lceil \log n_k \rceil$

Example for results: a flexible manufacturing system

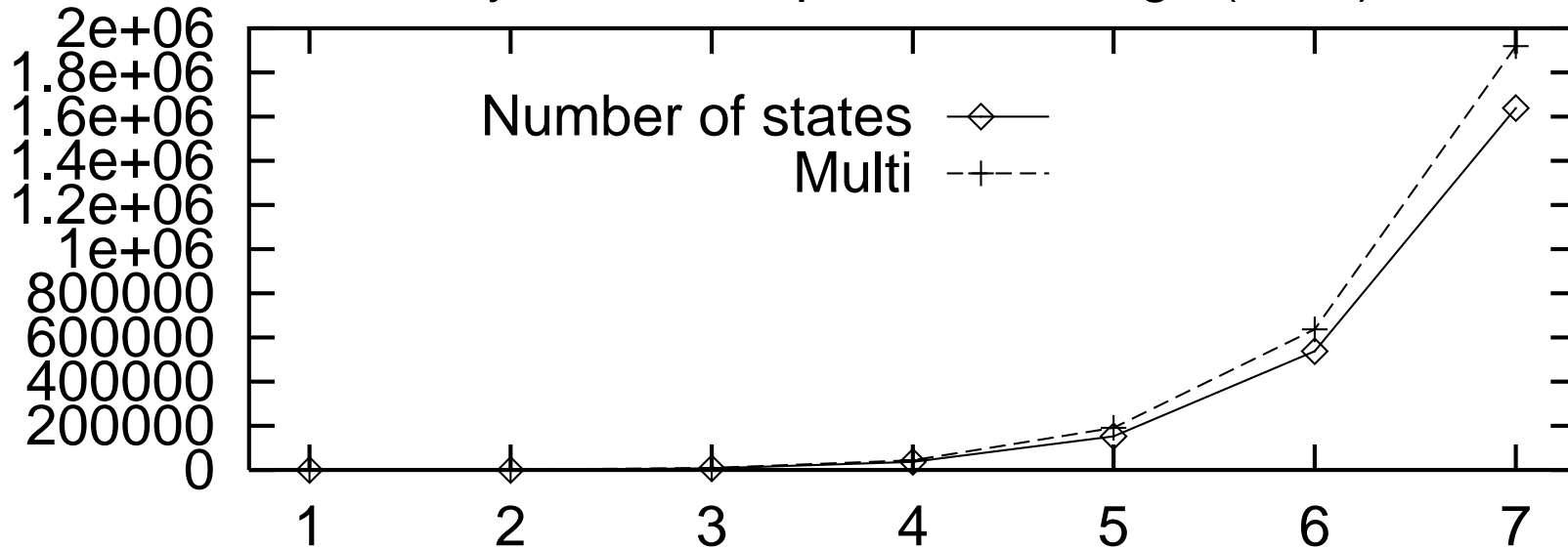


Results for state space generation (as a function of T)



Results for compression and state search (as a function of T)

Bytes for compressed storage (FMS)



Seconds to search 100,000 states (FMS)

