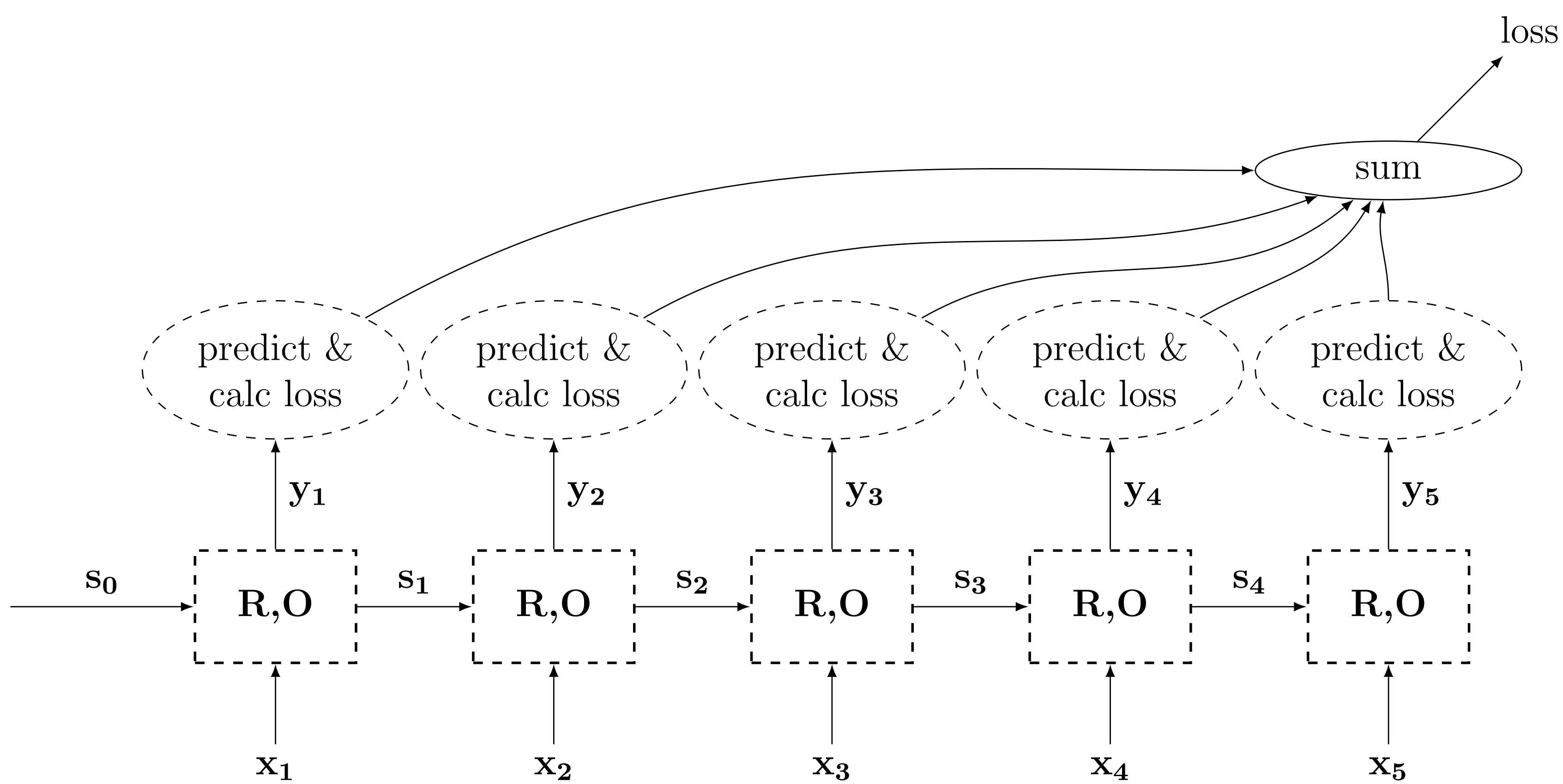
Extracting Automata from Recurrent Neural Networks Using Queries and Counterexamples

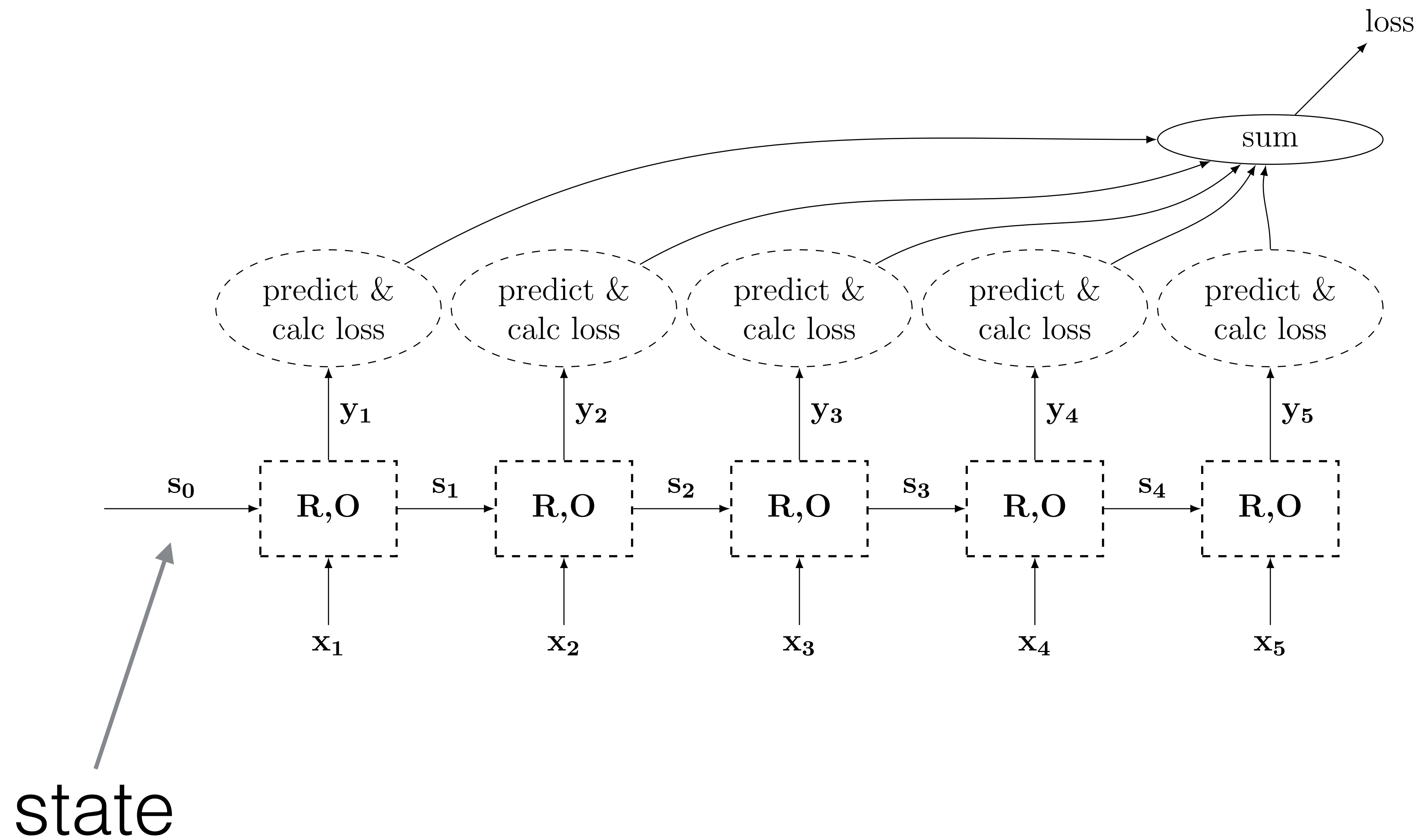
Gail Weiss[1], Yoav Goldberg[2], and Eran Yahav[1]

RNN acceptors as State Machines

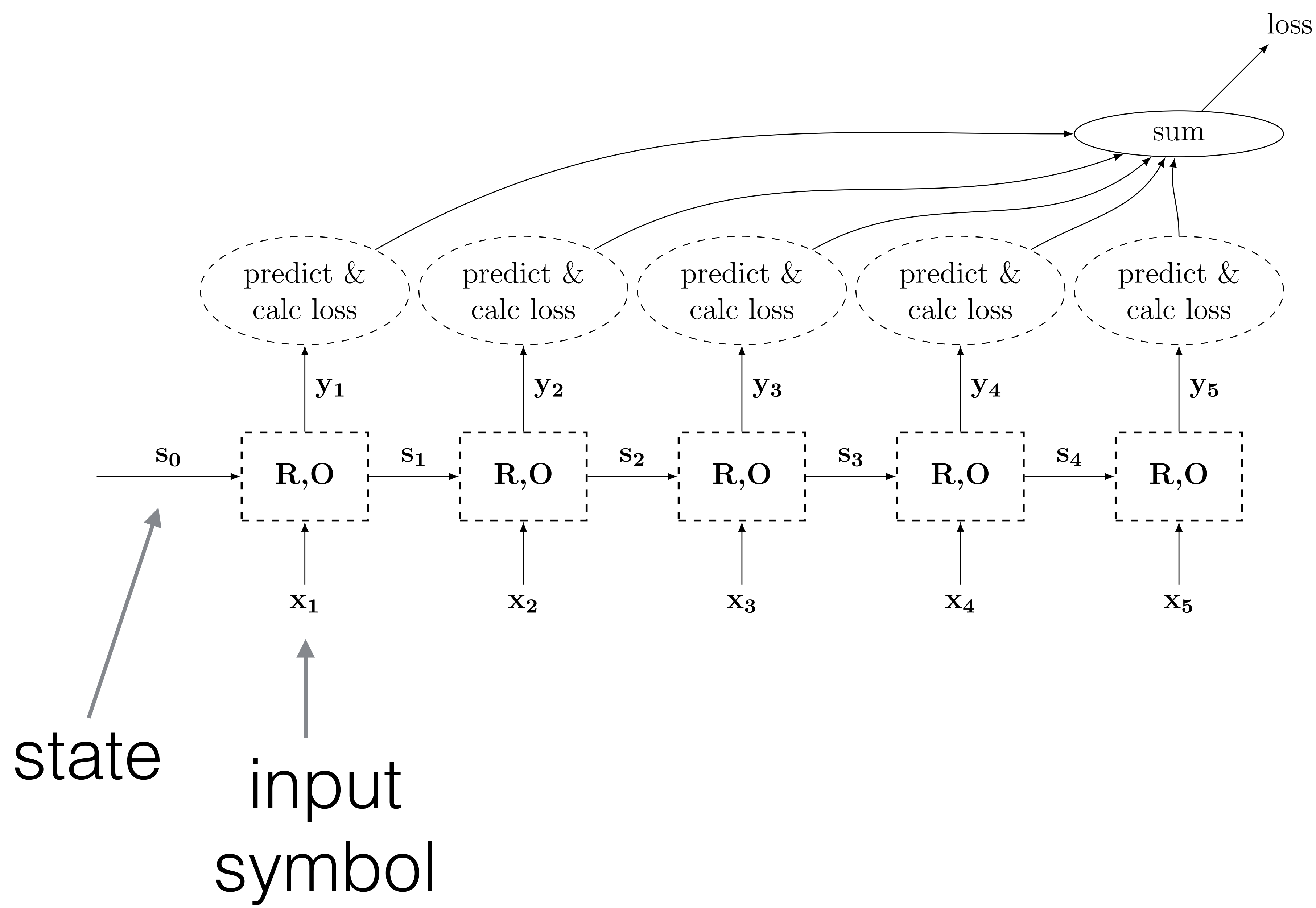
RNN acceptors as State Machines



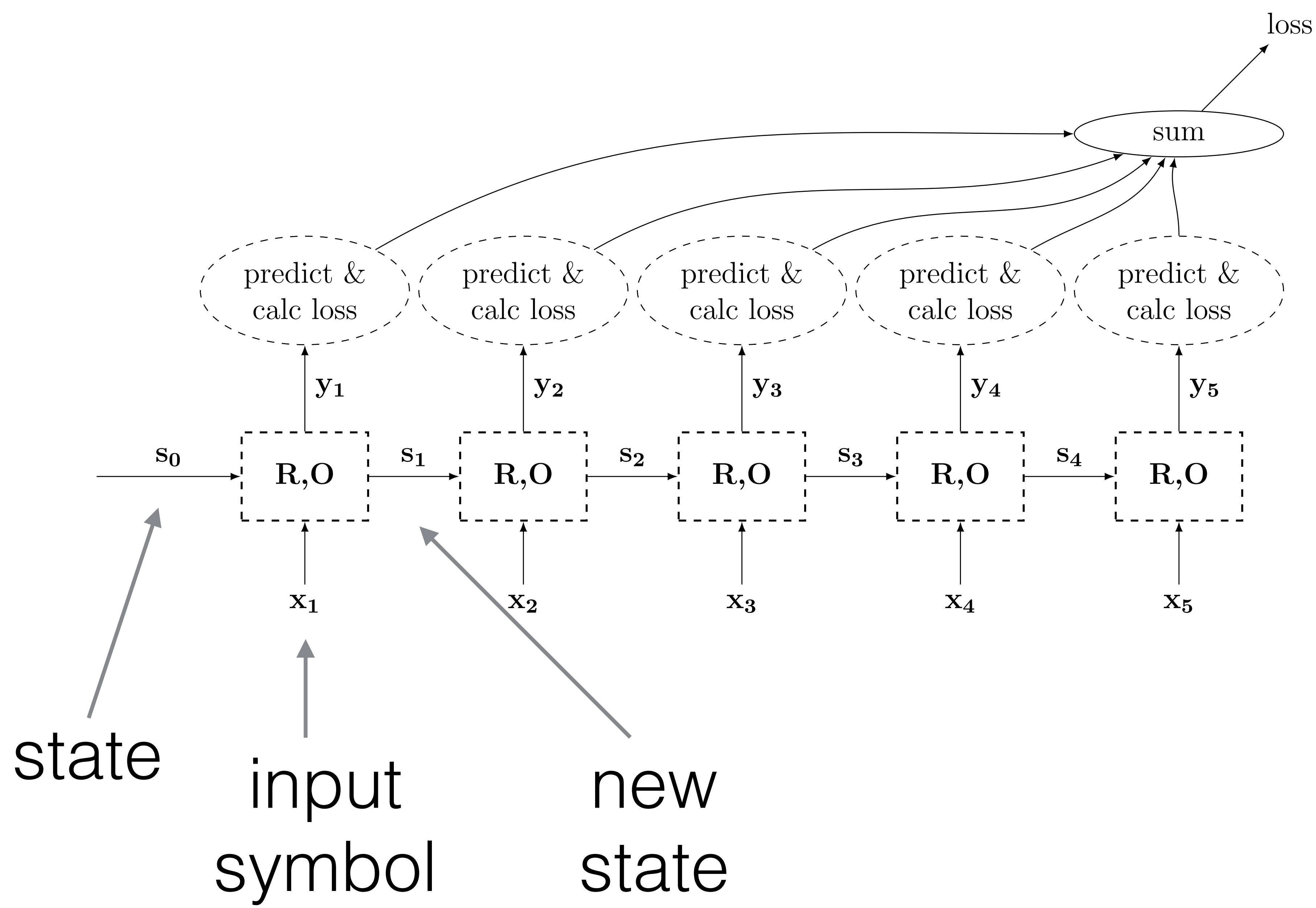
RNN acceptors as State Machines



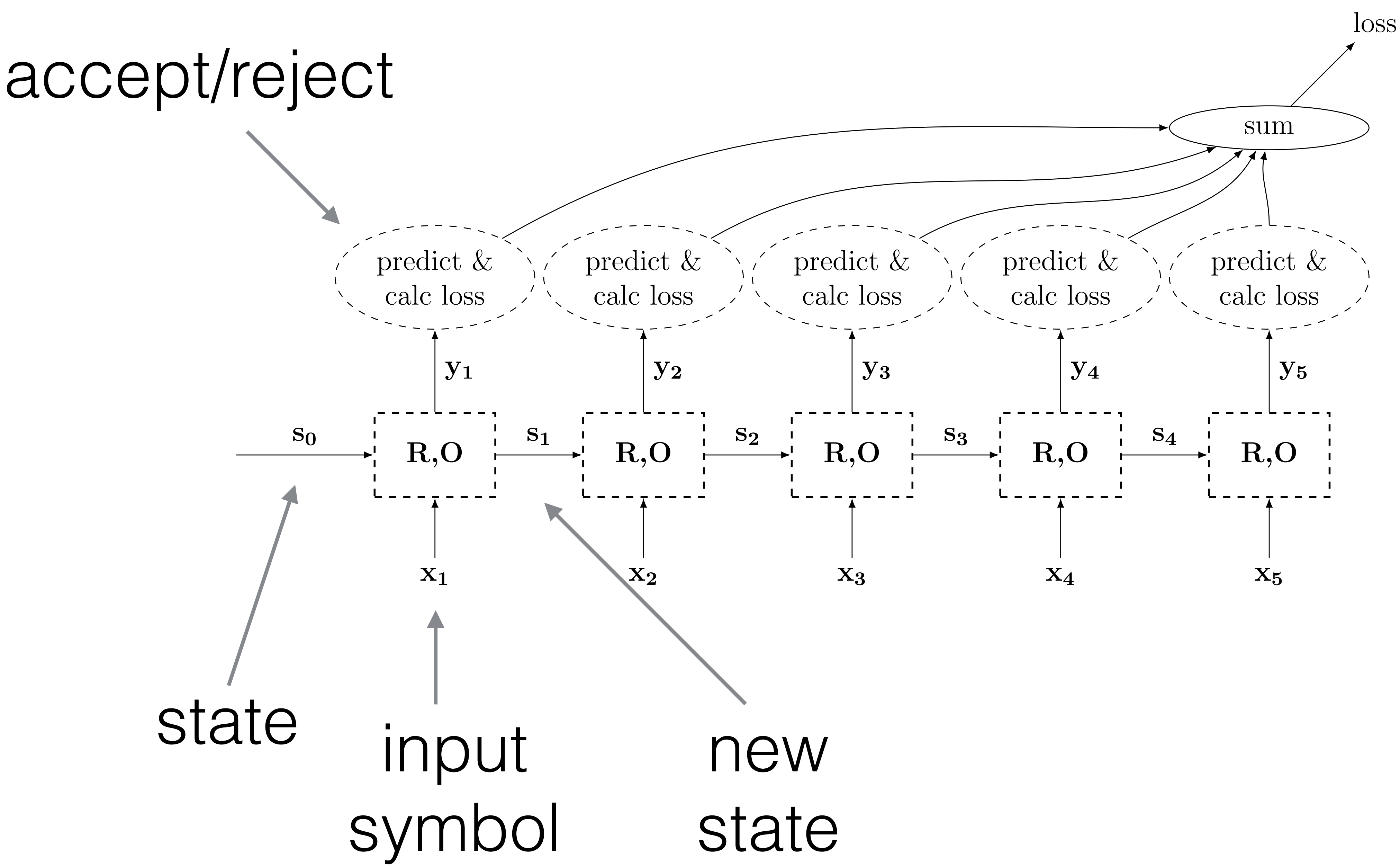
RNN acceptors as State Machines



RNN acceptors as State Machines

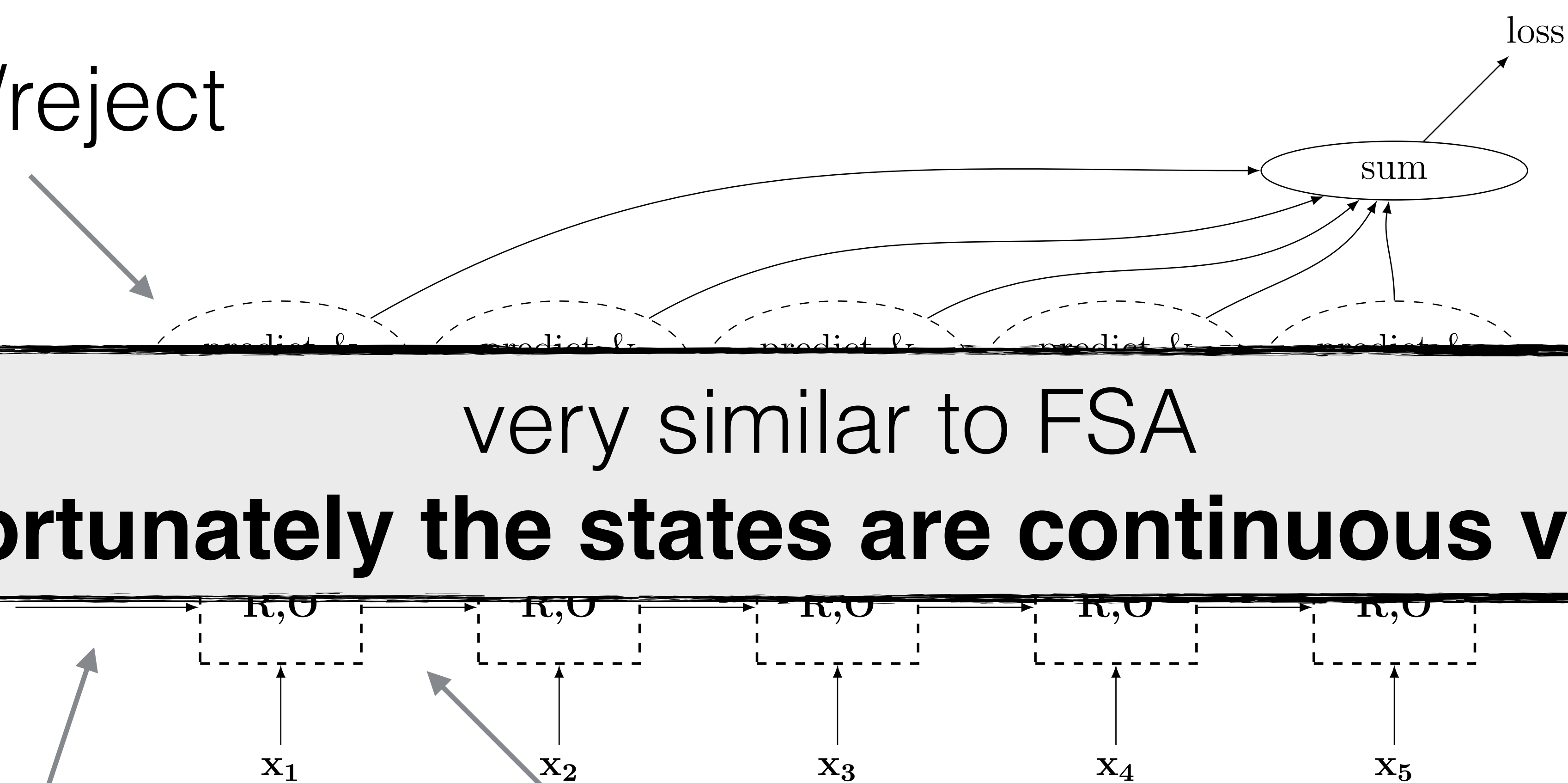


RNN acceptors as State Machines



RNN acceptors as State Machines

accept/reject



very similar to FSA

unfortunately the states are continuous vectors

state

input symbol

new state

INFORMATION AND COMPUTATION **75**, 87–106 (1987)

Learning Regular Sets from Queries and Counterexamples*

DANA ANGLUIN

Department of Computer Science, Yale University,
P.O. Box 2158, Yale Station, New Haven, Connecticut 06520

Learning
Finite State Automata

- **L* algorithm**

- FSAs are learnable from "**minimally adequate teacher**"
    - **Membership queries**

      "does this word belong in the language?**"**
    - **Equivalence queries**

      "does this automaton represent the language?"

Game Plan

- Train an RNN
- Use it as a Teacher in the L* algorithm
- L* learns the FSA represented by the RNN

RNN as Minimally Adequate Teacher

Membership Queries

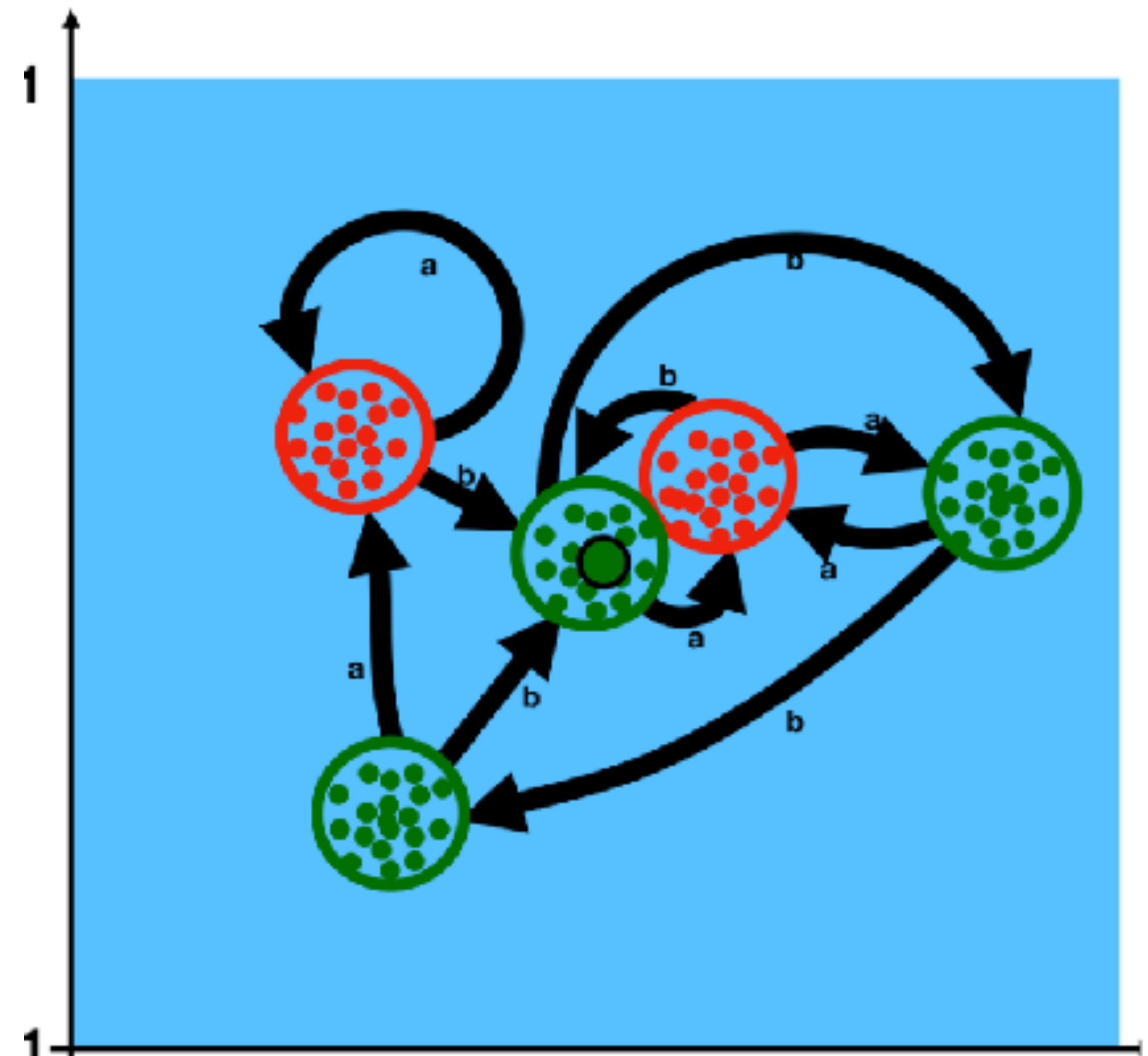
Easy. Just run the word through the RNN.

Equivalence Queries

Hard. Requires some trickery.

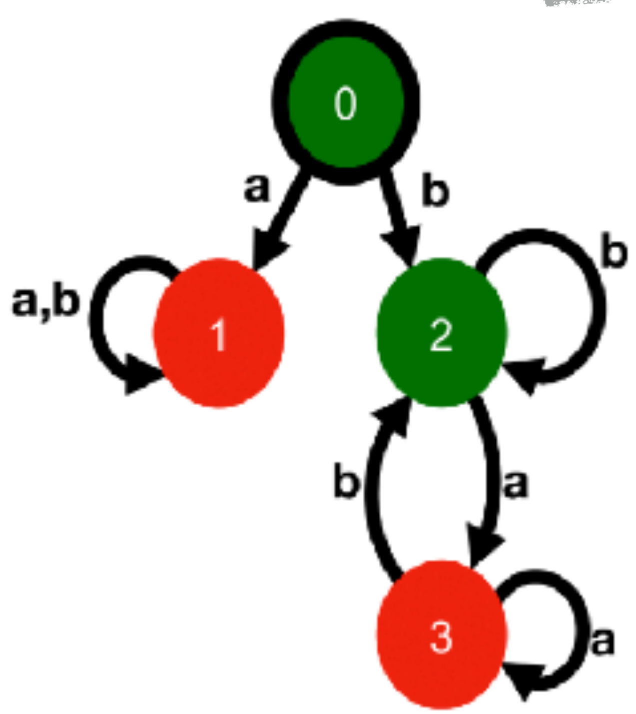
Answering Equivalence Queries

- Map RNN states to discrete states, forming an FSA abstraction of the RNN.

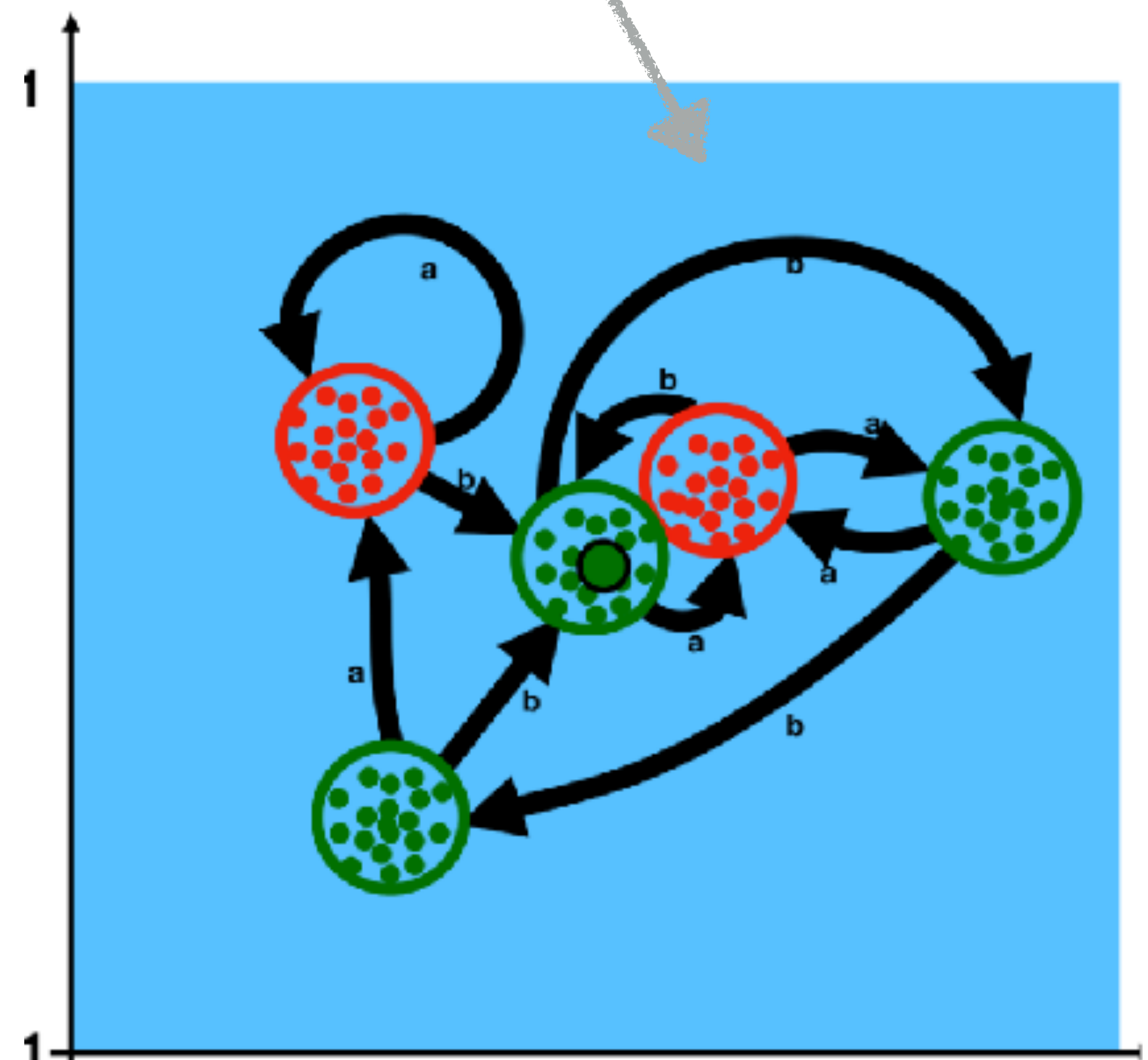


Answering Equivalence Queries

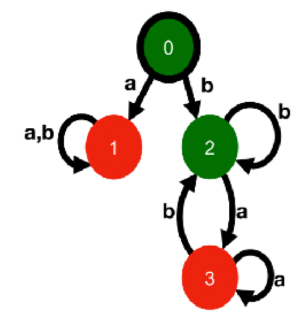
- Compare L* **Query FSA** to **RNN-Abstract-FSA**.

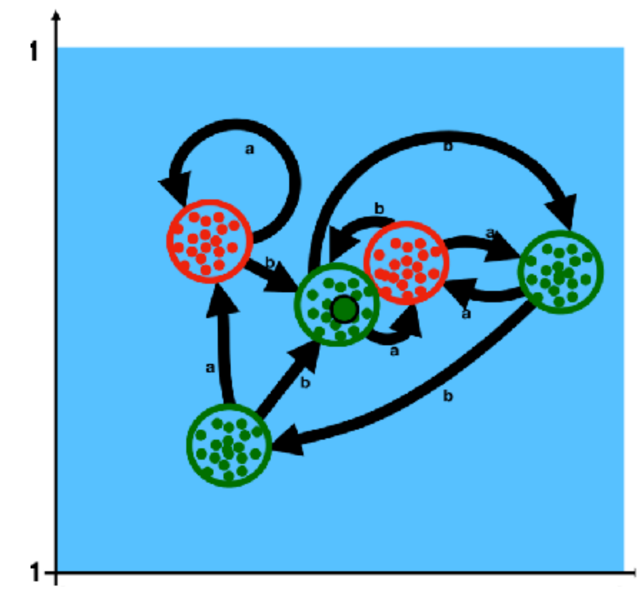


=??



Answering Equivalence Queries





- **Conflict**?
  - Maybe state-mapping is wrong.
    If so: **refine the mapping**.
  - Maybe L* FSA is wrong.
    If so: **return a counter example**.

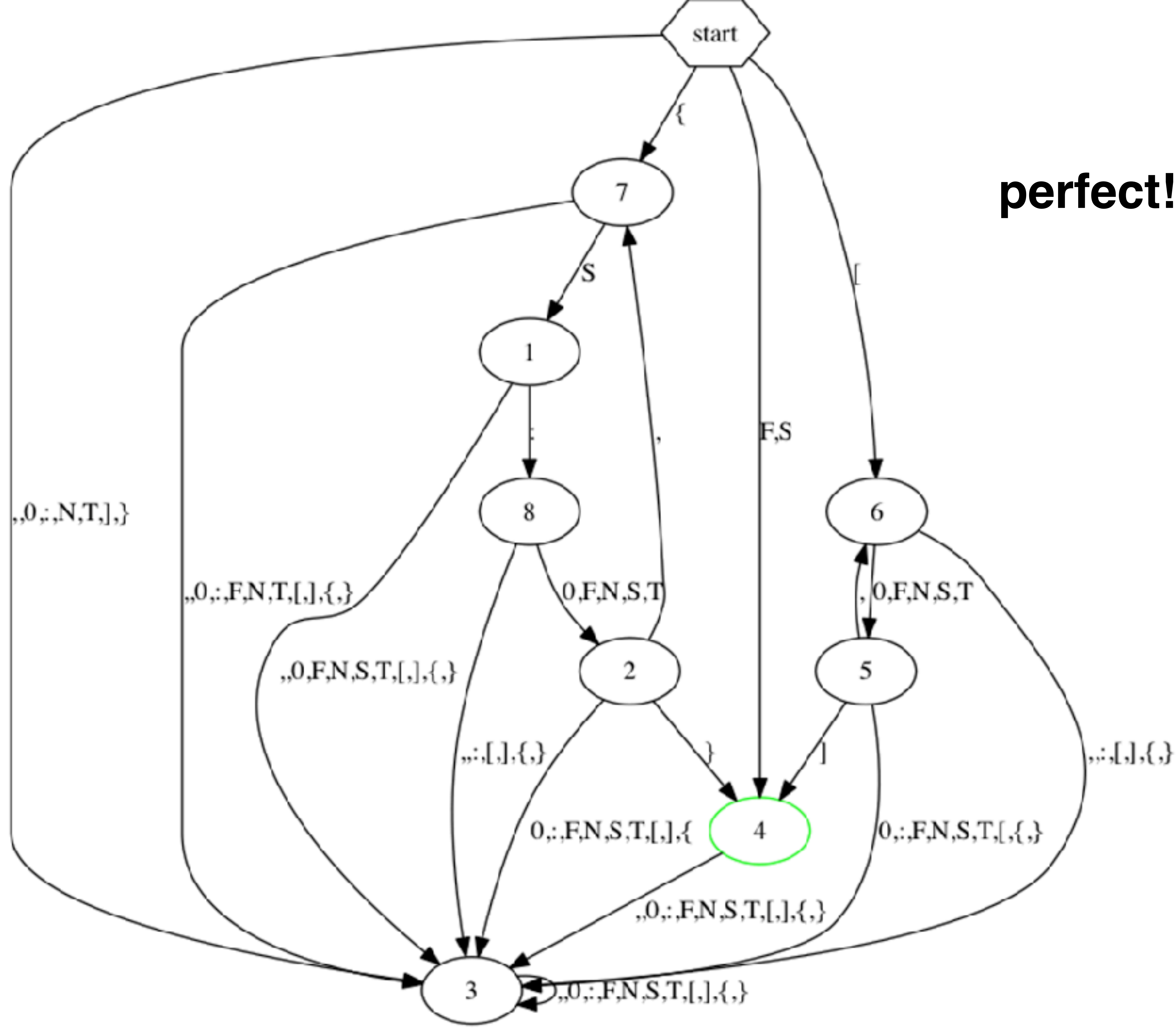
Some Results

- **Many random FSAs:**
  - 5 or 10 states, alphabet sizes of 3 or 5
- LSTM/GRU with 50, 100, 500 dimensions.
- The FSAs were **learned well** by LSTM / GRU
- And **recovered well** by L*.

"lists or dicts"

- `F`
- `S`
- `[F,S,0,F,N,T]`
- `{S:F,S:F,S:0,S:T,S:S,S:N}`

alphabet: `F S 0 N T , : { } [ ]`



start
{
7
S
1
:
8
,
F,S
[
6
,,0,:,N,T,],}
,,0,:,F,N,T,[,],{,}
,,0,F,N,S,T,[,],{,}
0,F,N,S,T
,
0,F,N,S,T
2
5
,,:,[,],{,}
}
]
,,:,[,],{,}
0,:,F,N,S,T,[,],{
4
0,:,F,N,S,T,[,{,}
,,0,:,F,N,S,T,[,],{,}
3
,,0,:,F,N,S,T,[,],{,}

perfect!

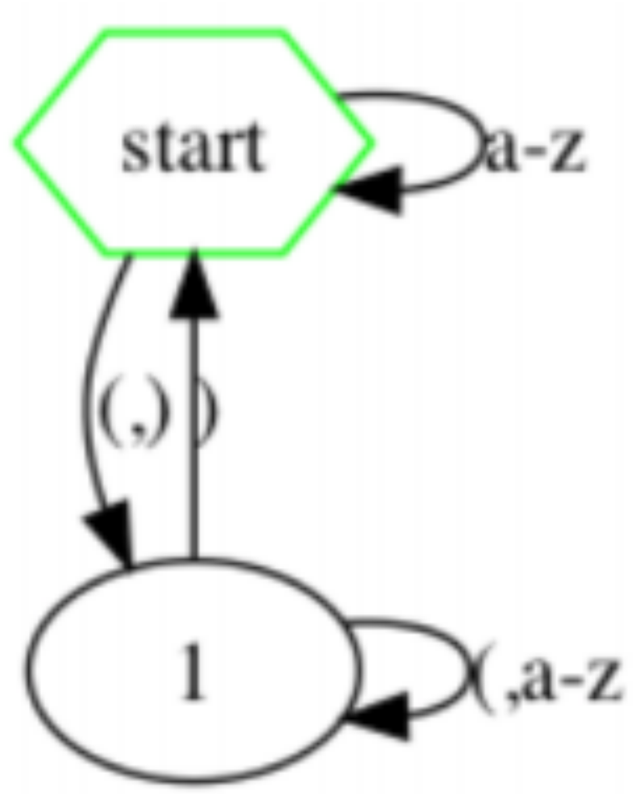
Balanced Parenthesis

`(a((ejka((acs))(asdsa))djljf)kls(fjkljklkids))`

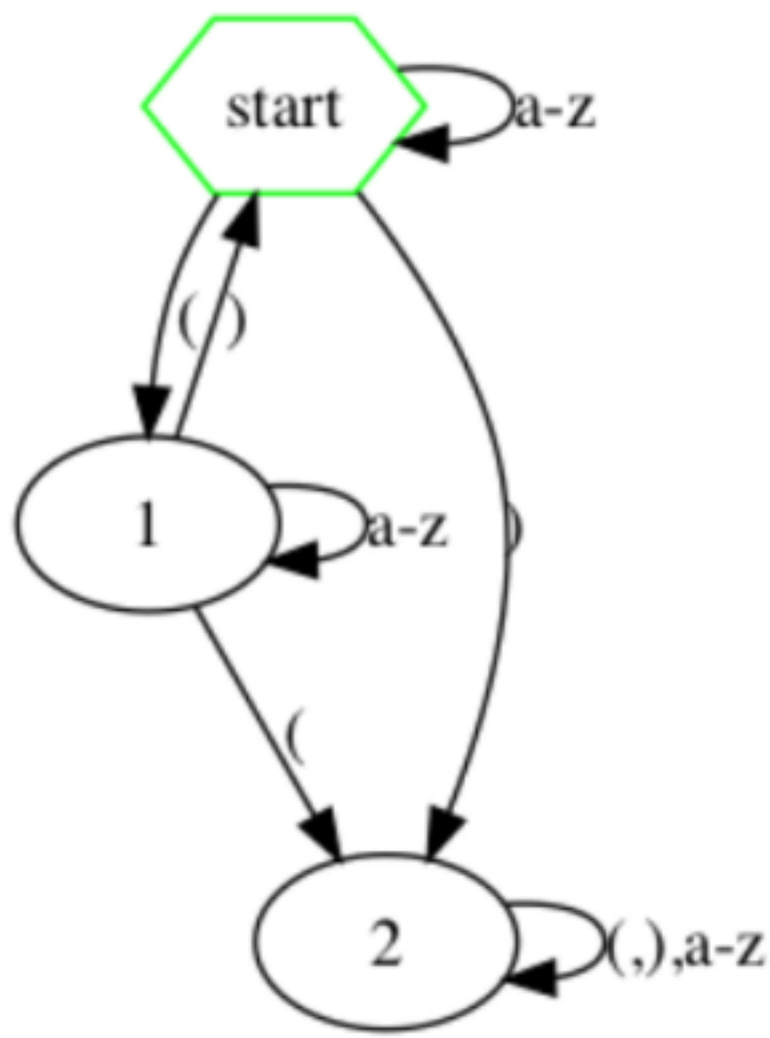
alphabet: `a-z ( )`

nesting level up to 8.

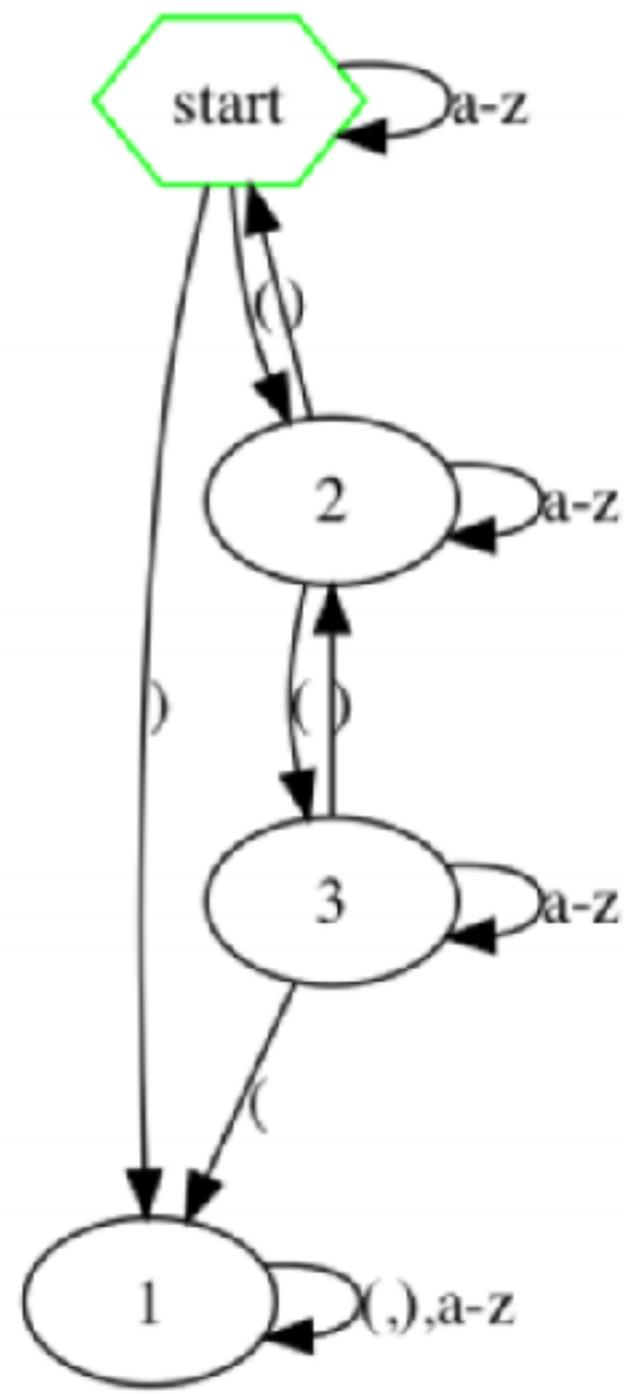
Balanced Parenthesis



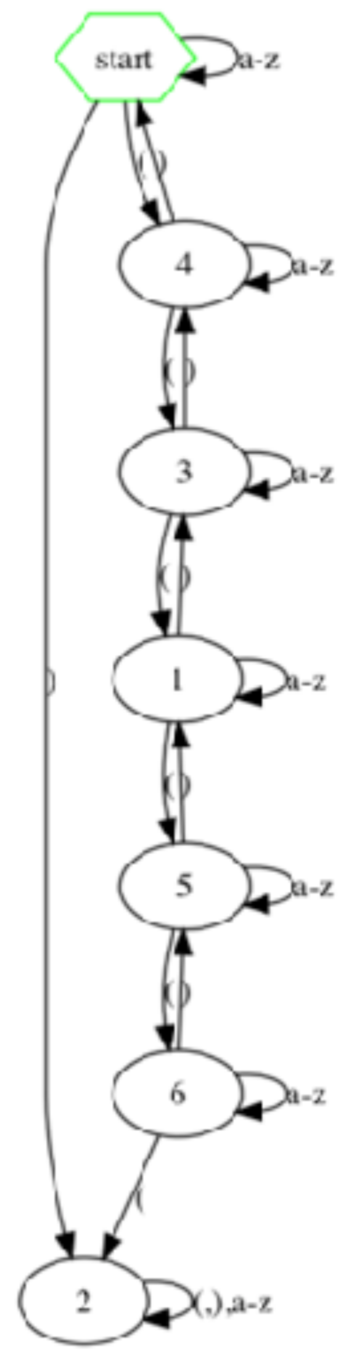
Balanced Parenthesis



Balanced Parenthesis



Balanced Parenthesis

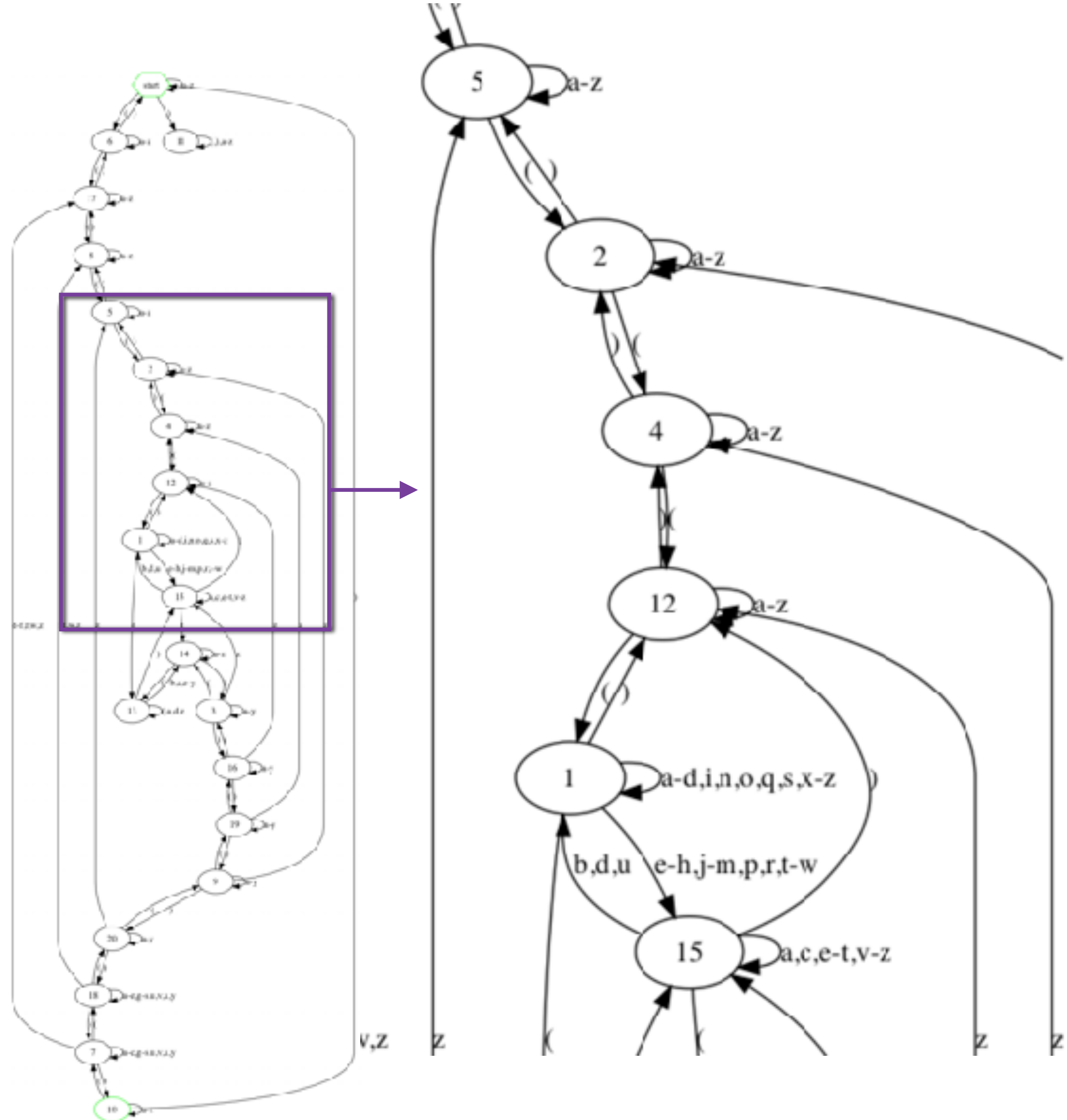


Balanced Parenthesis

final automaton:

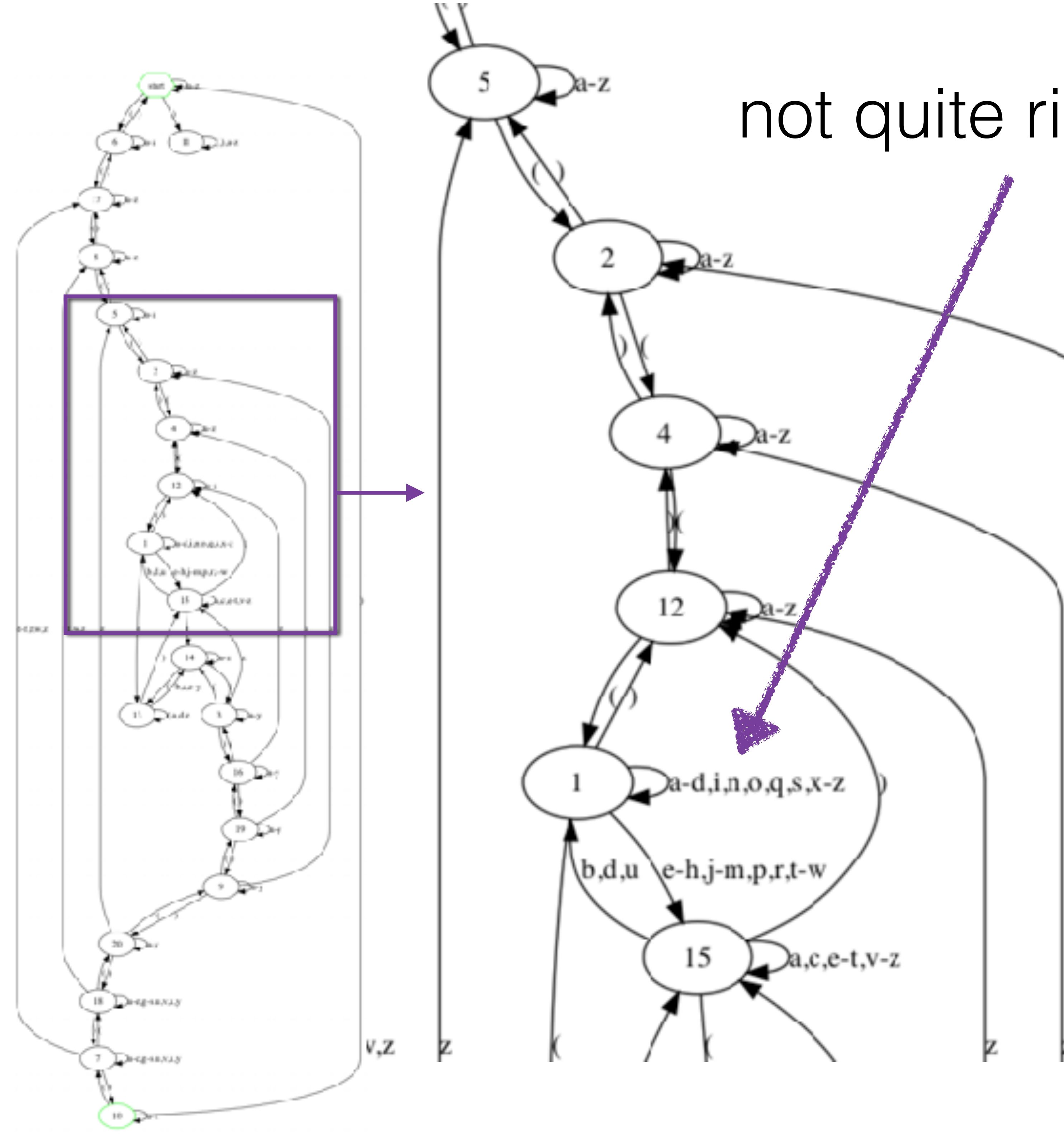
Balanced Parenthesis

final automaton:



Balanced Parenthesis

final automaton:



not quite right

"Emails"

- `bla12@abc.com, ahjlkoo@jjjgs.net`

`[a-z][a-z0-9]*@[a-z0-9]+\.(com|net|co\.[a-z][a-z])`

"Emails"

- `bla12@abc.com, ahjlkoo@jjjgs.net`

```
[a-z][a-z0-9]*@[a-z0-9]+\.(com|net|co\.[a-z][a-z])
```

20,000 positive examples
20,000 negative examples
2,000 examples dev set

"Emails"

- `bla12@abc.com, ahjlkoo@jjjgs.net`

`[a-z][a-z0-9]*@[a-z0-9]+\.(com|net|co\.[a-z][a-z])`

20,000 positive examples
20,000 negative examples
2,000 examples dev set

LSTM has 100% accuracy on both train and dev (and test)

"Emails"

the extraction algorithm did not converge.
we stopped it when it reached over 500 states.

some examples it found:

25.net

5x.nem

2hs.net

LSTM has 100% accuracy on both train and dev (and test)

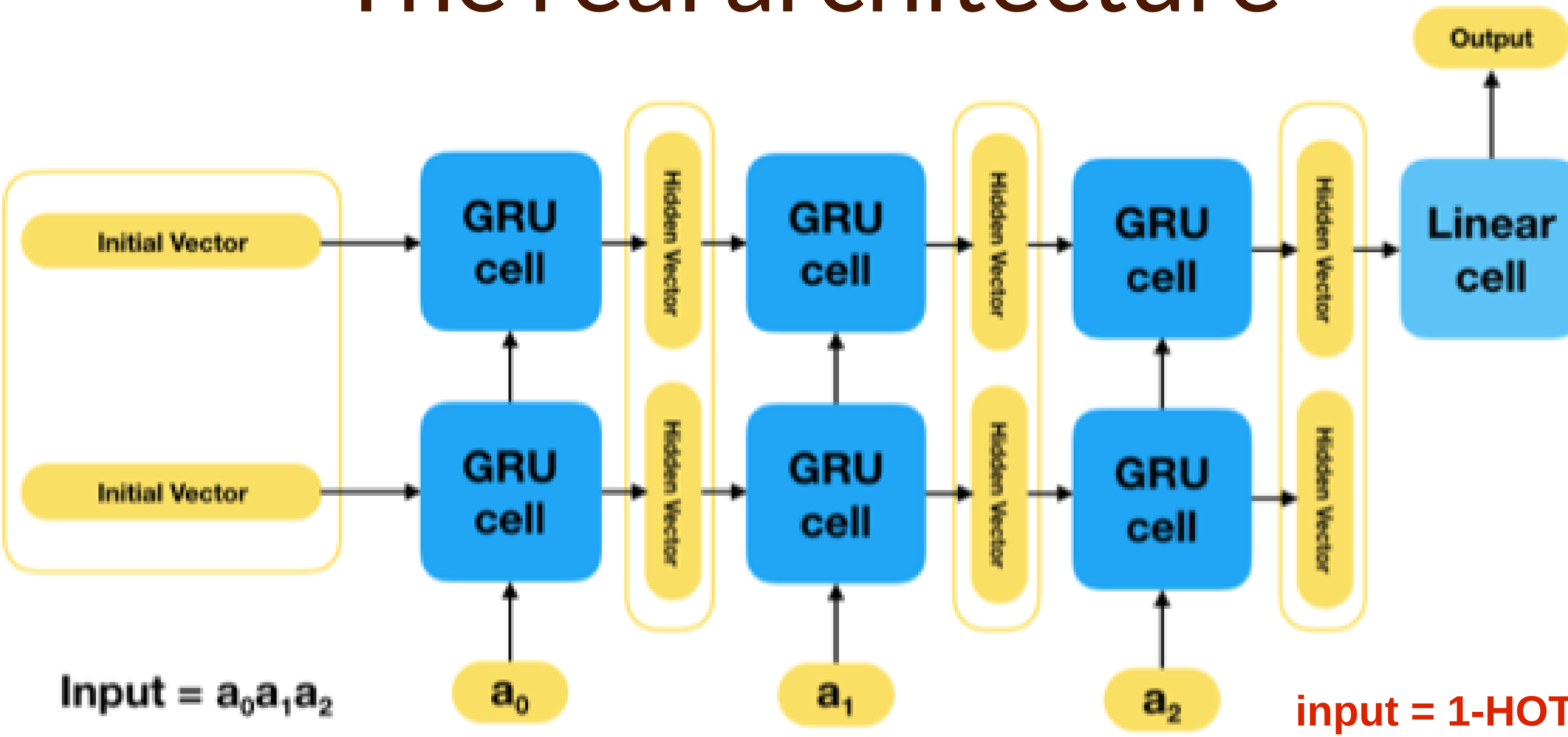
- **We can extract FSAs from RNNs**
  - ... if the RNN indeed captured a regular structure
  - ... and in many cases the representation captured by the RNN is much more complex (and wrong!) than the actual concept class.

- **Much more to do:**
    - scale to larger FSAs and alphabets
    - scale to non-regular languages
    - apply to "real" language data
    - ....

Hot points

- The real neural architecture
- L*
- Equivalence query
  - Come confronto l'ipotesi di fatta da L* con l'automa codificato in RNN?
  - Estrazione di un DFA da una RNN *guidata* dall'ipotesi di L*
  - Refinement -> parallel traversal

The real architecture



input = 1-HOT-vectors

Illustration of an RNN based network, with two layers and a linear classifier, unrolled on an input sequence of length 3. At each iteration, ***the concatenation of each layer's state vectors can be seen as the current network state****, with* ***the network implicitly defining a deterministic transition function between these states****. The linear cell is applied to the final output of the top layer of the network to give its classification on the entire input sequence.*

Extracting DFA from RNN-> *partition function* [Omlin and Giles96]

- p: S → N

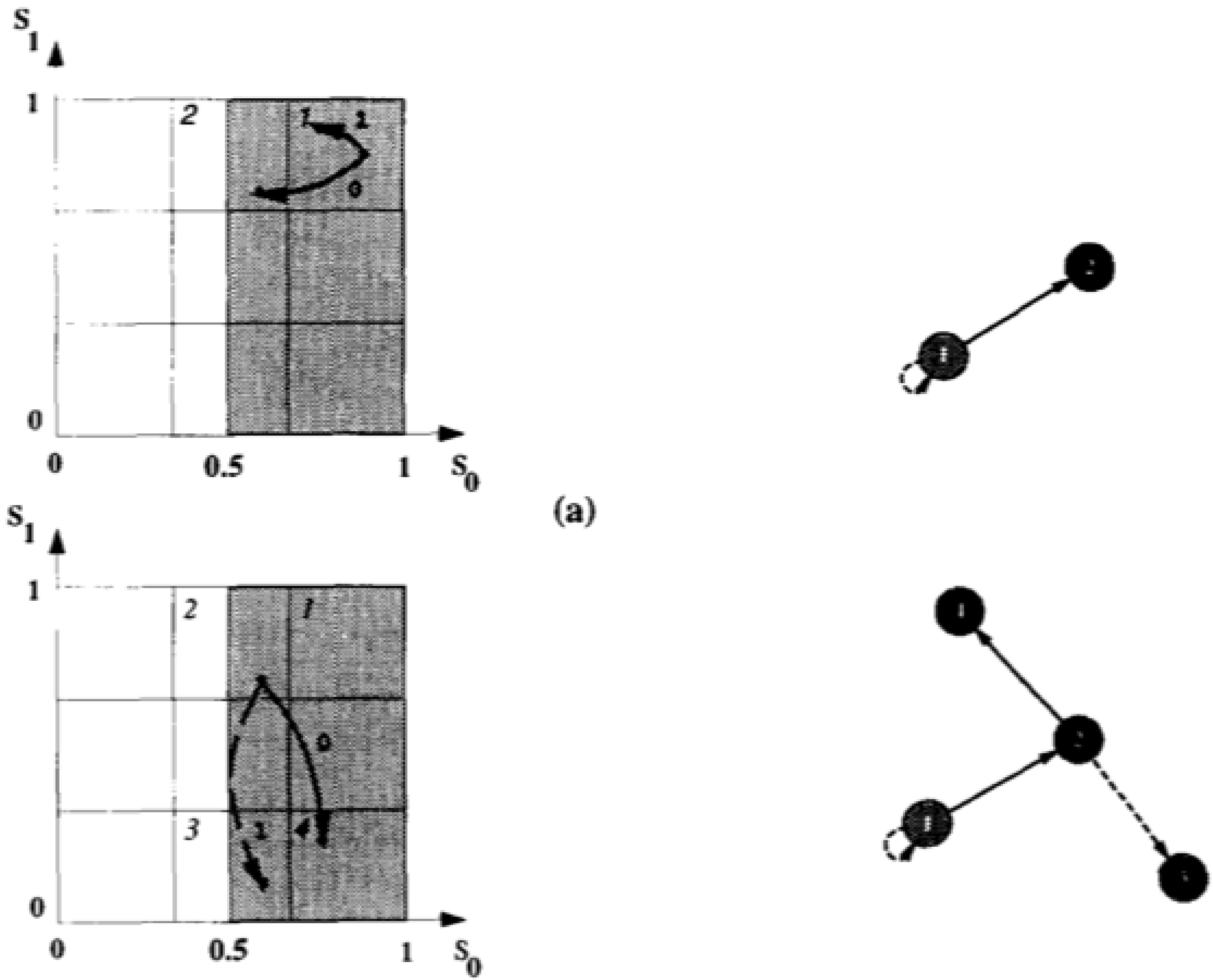
- *... in which **every abstracted state is an entire partition from p** and the **transitions between abstracted states** and their classifications are obtained by a **single sample of the continuous values in each such partition.*** -> Quantisation (!)

- Experimental clusters formation: *Omlin and Giles showed that RNN-states tend to cluster in small areas in the network state space, and—along with an assumption on the continuity of the network's behavior—concluded that it was safe to cluster like-valued state vectors together as one state.*

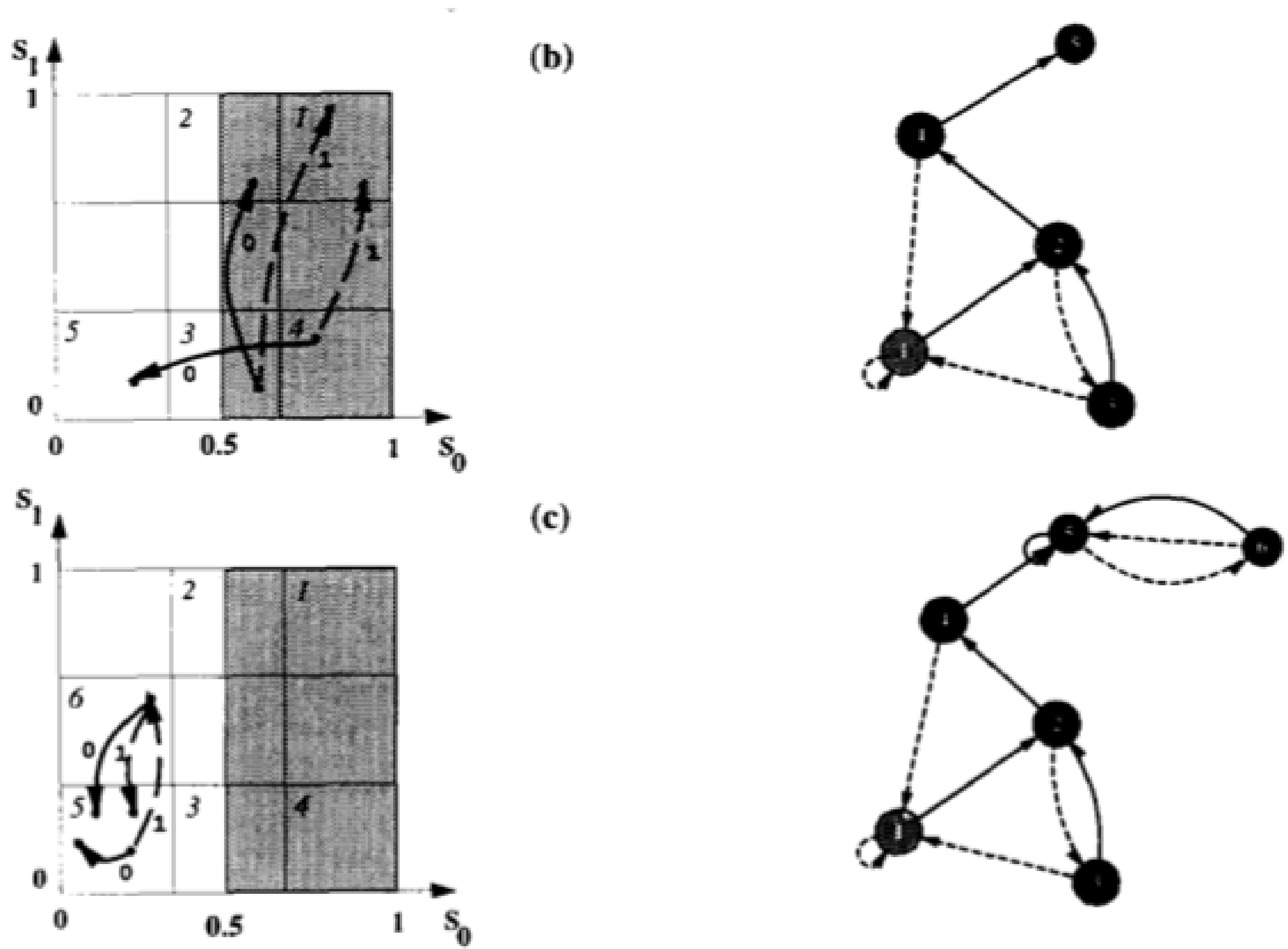
Extracting DFA from RNN-> *partition function* [Omlin and Giles96]

$Q, F, \delta \leftarrow \emptyset$
$New \leftarrow \{h_0\}$
while $New \neq \emptyset$ **do**
    $h \leftarrow$ pick and remove from New
    $q \leftarrow p(h)$
    if $q \notin Q$ **then**
        $Q \leftarrow Q \cup \{q\}$
        if $f_N(h) = Acc$ **then** $F \leftarrow F \cup \{q\}$
        $\boxed{\textbf{for } \sigma \in \Sigma \textbf{ do}}$
            $h' \leftarrow g_N(h, \sigma)$
            $\delta \leftarrow \delta \cup \{((q, \sigma), p(h'))\}$
            $New \leftarrow New \cup \{h'\}$
        end
    end
end

Extracting DFA from RNN-> *partition function* [Omlin and Giles96]



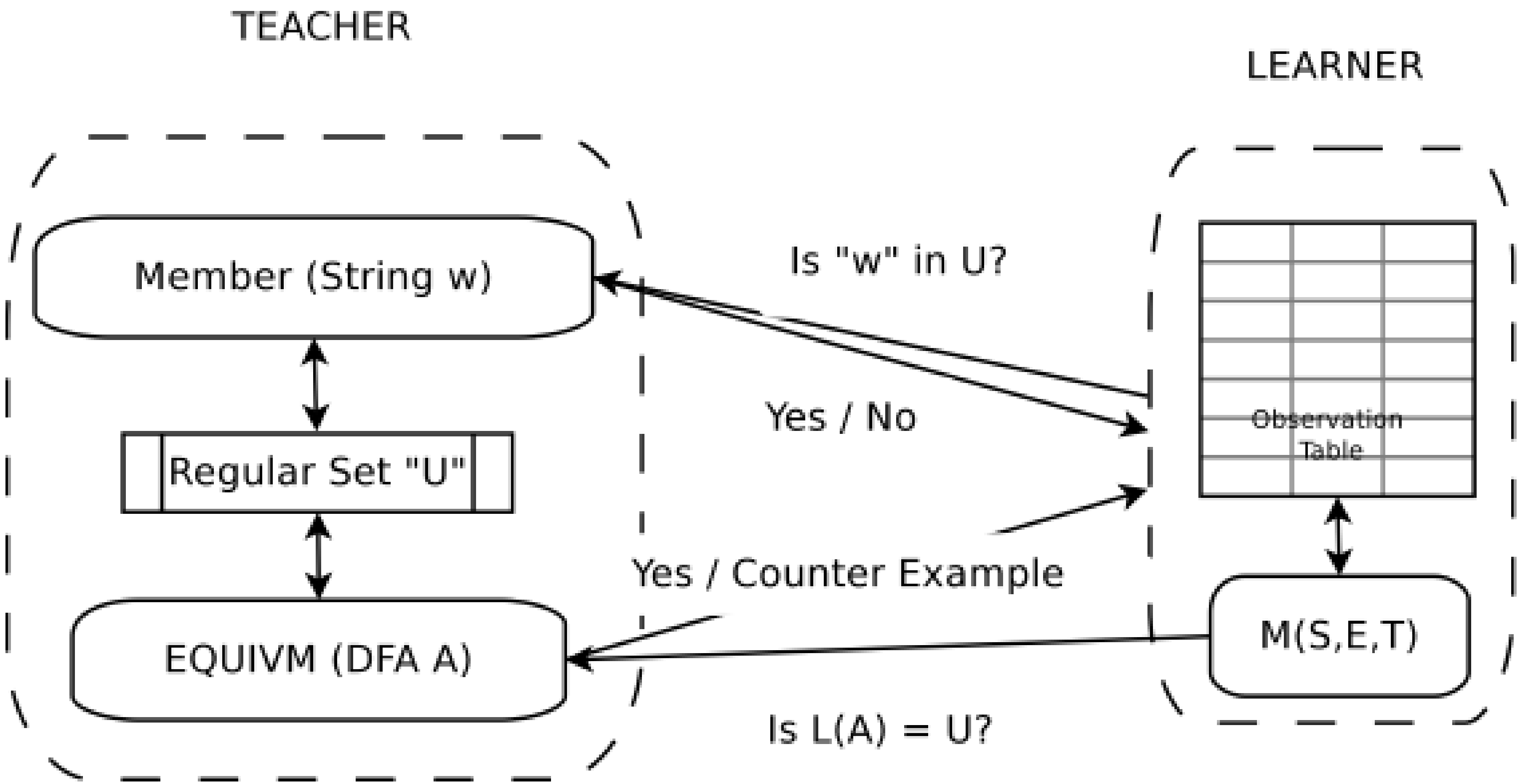
Extracting DFA from RNN-> *partition function* [Omlin and Giles96]



Extracting DFA from RNN-> *partition function* [Omlin and Giles96]

- BFS exploration
- Blind unrolling
- *Without an **oracle**'s guidance, minimizing an automaton is impossible before unrolling it in its entirety, and so this is true even if, when minimized, the representative automaton for the network is quite small.*

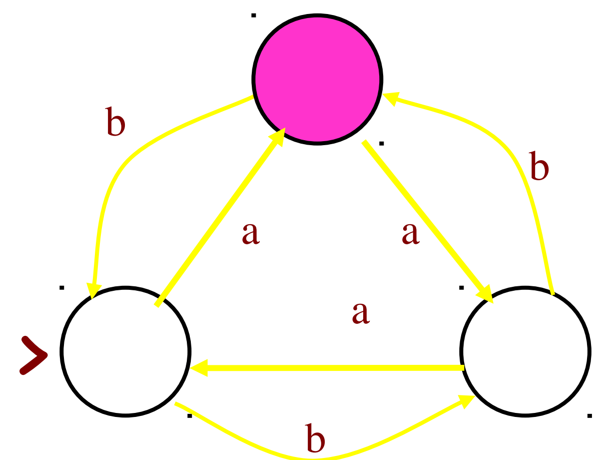
L*


TEACHER
LEARNER
Member (String w)
Regular Set "U"
EQUIVM (DFA A)
Is "w" in U?
Yes / No
Yes / Counter Example
Is L(A) = U?
Observation Table
M(S,E,T)

L*

- Key idea is to represent the DFA using a experiments/states/transitions table.

	Experiments	λ	a
States	λ	□	■
	a	■	□
	b	□	□
Transitions	aa	□	□
	ab	□	■
	ba	□	■
	bb	■	□



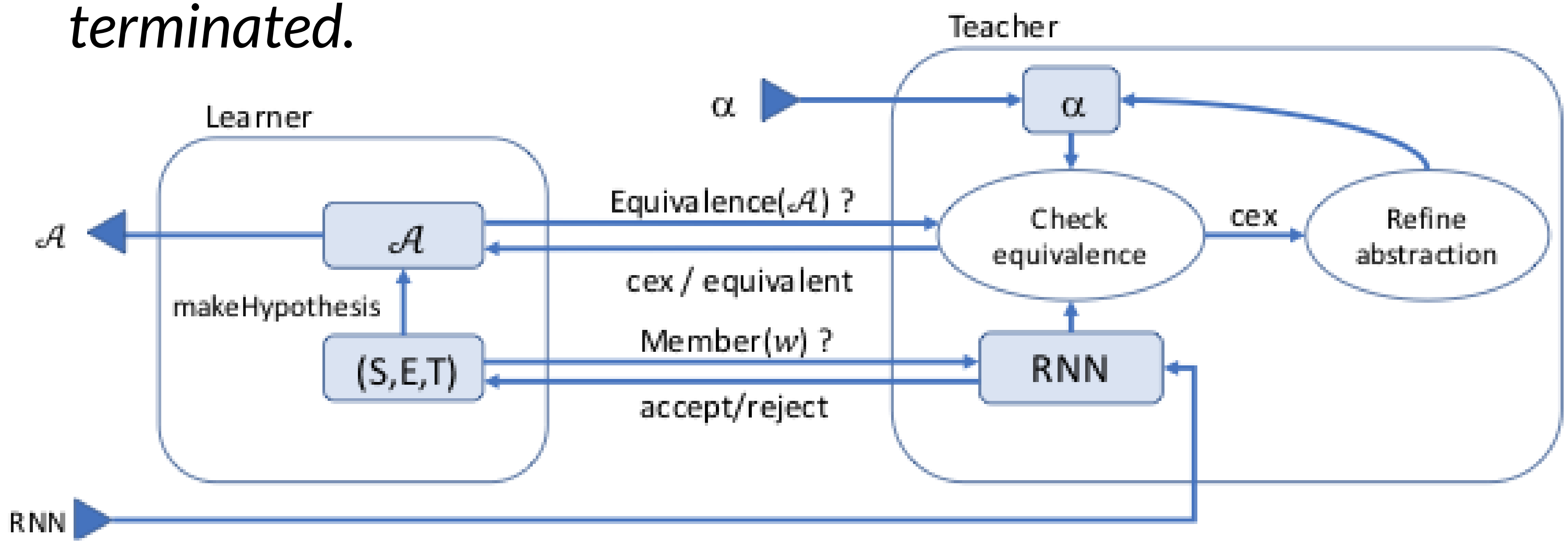
- Closure
- Consistency

L*

```
S ← {ε}, E ← {ε}
foreach (s ∈ S), (a ∈ Σ), and (e ∈ E) do
    T[s, e] ← Member(s · e)
    T[s · a, e] ← Member(s · a · e)
end
repeat
    while (s_new ← Closed(S, E, T) ≠ ⊥) do
        Add(S, s_new)
        foreach (a ∈ Σ, e ∈ E) do T[s_new · a, e] ← Member(s_new · a · e)
    end
    𝒜 ← MakeHypothesis(S, E, T)
    cex ← Equivalence(𝒜)
    if cex = ⊥ then
        return 𝒜
    else
        e_new ← FindSuffix(cex)
        Add(E, e_new)
        foreach (s ∈ S, a ∈ Σ) do
            T[s, e_new] ← Member(s · e_new)
            T[s · a, e_new] ← Member(s · a · e_new)
        end
    end
until until
```

RNN as teacher

The learner and teacher iteratively refine their automatons and abstractions, generating two series of automata ... until the automata either converge, or the interaction is terminated.



Check Equivalence

- Key idea: **refine** the partition of the RNN by using the L* hypothesis
- States
  - L-states by L* hypotheses
  - A-states by the Abstraction (partition)
  - R-states by the RNN
- *The key intuition to our approach is the fact that [DFA L*-hypothsis] is minimal, and so each state in the [DFA Abstraction] should — if the two automatons are equivalent — be equivalent to exactly one state in the [DFA L*-hypothsis]*
- **Conflicts**
  - **Clustering conflicts**: 1 A-state ≡ 2 L-states -> refine
  - **Classification conflicts**: $F_R(w) \neq F_L(w)$ -> counterexample

Abstraction Refinement

- Clusters resolution key idea: to identify clusters regions in the RNN one uses a SVM classifier
- *Intuitively, we would like to allocate a region around the R-state h that is large enough to contain other continuous R-states that behave similarly, yet still separate it from neighboring R-state (i.e., vectors in H) that behave differently. We achieve this by fitting an SVM classifier with RBF kernel to separate the single vector h from the set H*
- SVM aggiunge un nuovo stato nella nuova partizione differenziandolo da quelli già presenti nella precedente partizioe
- algoritmo complesso! ->

Brute force?

- Se uso un generatore bruto di controesempi che usa membership queries?
- Non funziona bene su automi complessi -> vedi articolo

Conclusioni

- Interessante: primo lavoro in cui si usa RNN come oracolo
- Simbolico <-> sub-simbolico
  - Refinement
- Funziona bene su linguaggi “abbastanza” complessi