Synthesizing Transducers from Complex Specifications

Anvay Grover, Rüdiger Ehlers, Loris D’Antoni

Verifying string manipulating programs:

A Decision Procedure for Path Feasibility of String Manipulating Programs with Integer Data Type

Taolue Chen[1], Matthew Hague
A
hnology, Czech Republic
nology, Czech Republic
ord, United Kingdom
ty, Sweden
Technology, Czech Republic

String Constraints with Concatenation and Transducers Solved Efficiently

High-Level Abstractions for Simplifying Extended String Constraints in SMT

Andrew Reynolds[1], Andres Nötzli[2(
Clark Barrett[2], and Cesare Tinell

CertiStr: A Certified String Solver

Shuanglong Kan
Department of Computer Science
Technische Universität Kaiserslautern
Kaiserslautern, Germany
shuanglong@cs.uni-kl.de

Anthony Widjaja Lin
Department of Computer Science
Technische Universität Kaiserslautern & MPI-SWS
Kaiserslautern, Germany
lin@cs.uni-kl.de

Philipp Rümmer
Department of Information Technology
Uppsala University
Uppsala, Sweden
philipp.ruemmer@it.uu.se

Micha Schrader
Department of Computer Science
Technische Universität Kaiserslautern
Kaiserslautern, Germany
schrader@rhrk.uni-kl.de

```
var x = goog.string.htmlEscape(name);
var y = goog.string.escapeString(x);
nameElem.innerHTML = '<button onclick= "viewPerson(\'' + y + '\')">' + x + '</button>';
```

Sanitize inputs

Prevent XSS attacks

```
function toUpperCase(inputStr) {
    ...
}
```

toUpperCase(Aba) → ABA

toUpperCase(ABA) → ABA

The toUpperCase **function is idempotent**

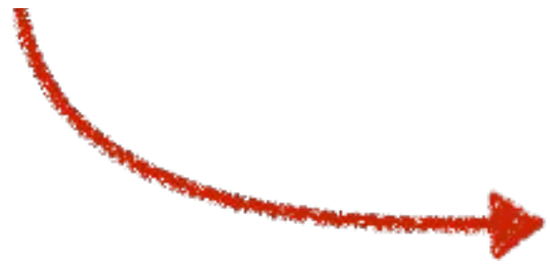
Suppose we want to prove:

v = toUpperCase(x)

A Decision Procedure for Path Feasibility of String Manipulating Programs with Integer Data Type

Taolue Chen[1], Matthew Hague[2], Jinlong He[3,6], Denghang Hu[3,6], Anthony Widjaja Lin[4], Philipp Rümmer[5], and Zhilin Wu[3,7,8(✉)]

')

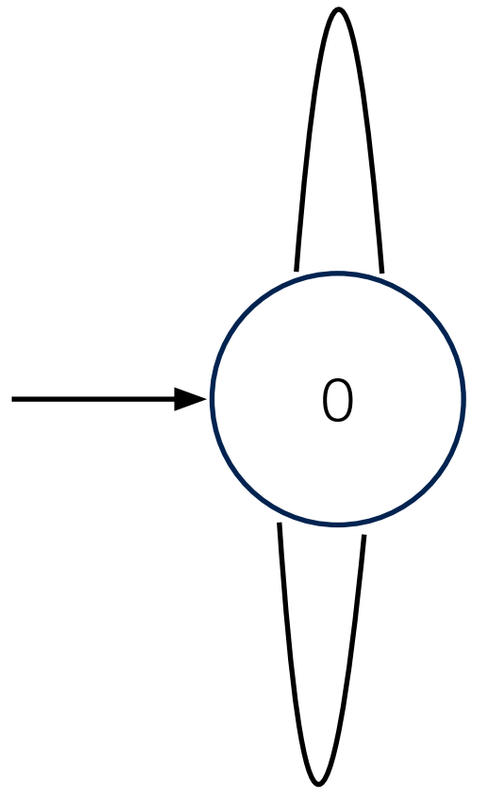


Ostrich: An SMT Solver for string constraints

But…

Ostrich models string to string functions as transducers!

toUpperCase



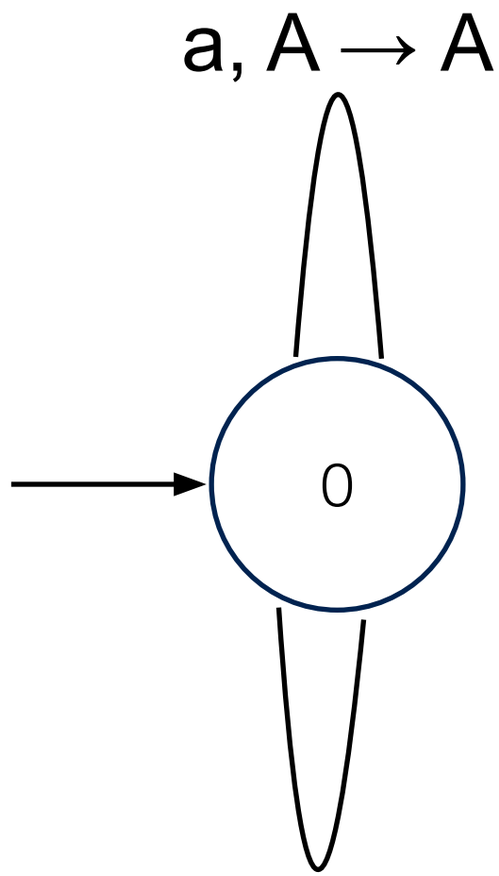
Why Transducers?

Decidability Properties

Closure under Composition

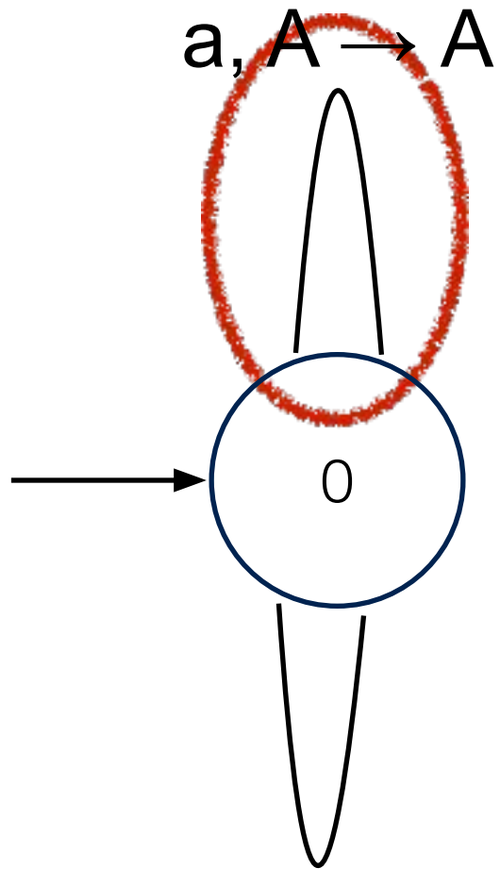
Equivalence Checking

toUpperCase(toUpperCase()) ≡ toUpperCase()



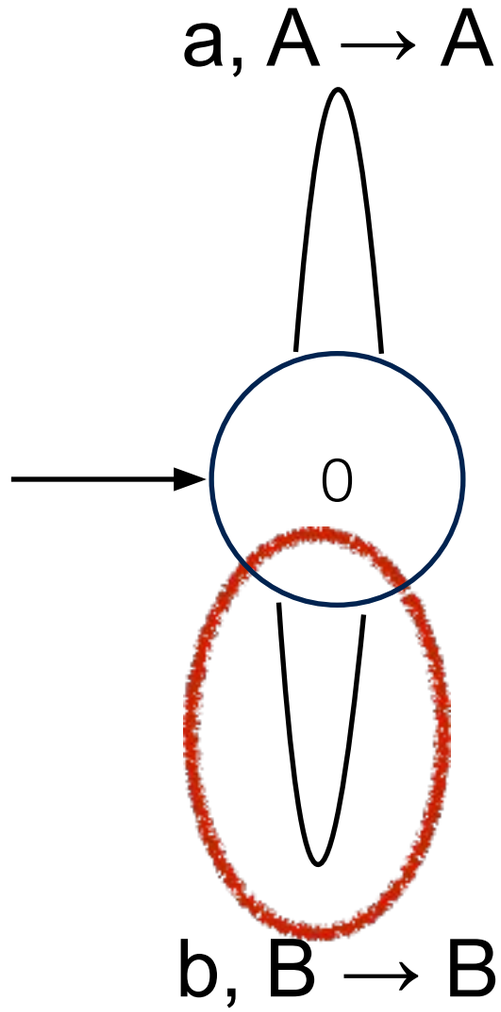
b, B → B

toUpperCase

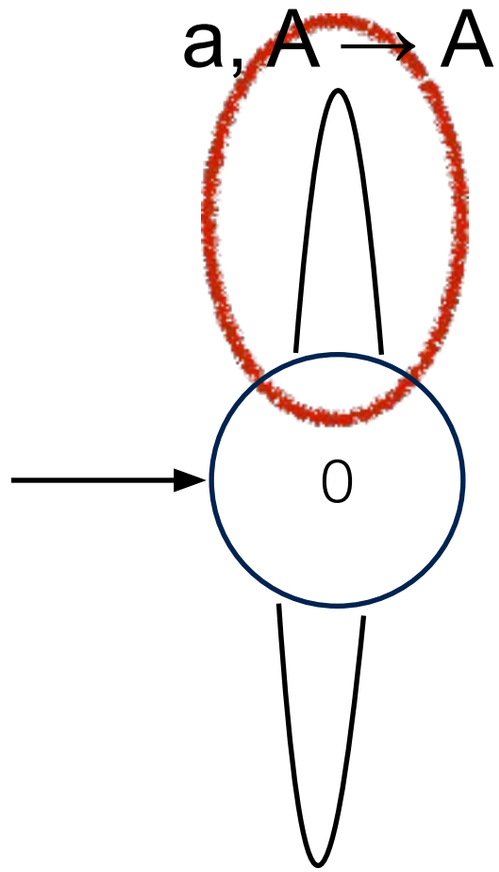


b, B → B

toUpperCase(aBa) → ABA

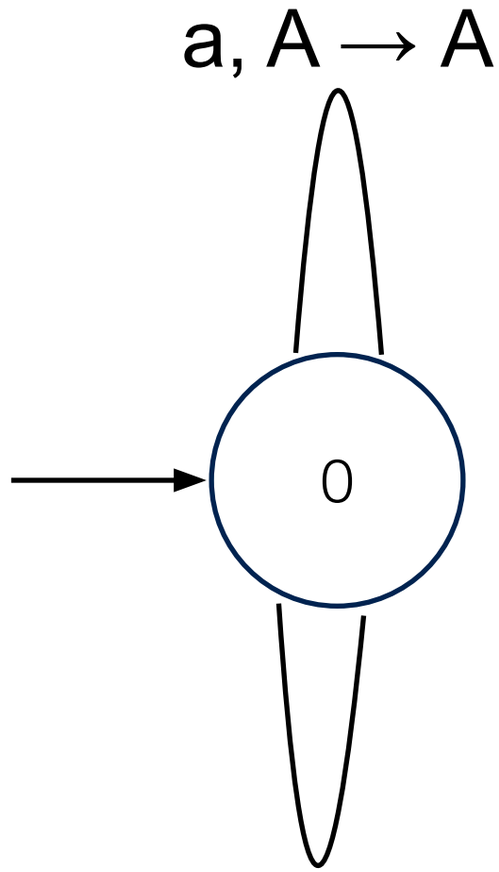


toUpperCase(aBa) → ABA



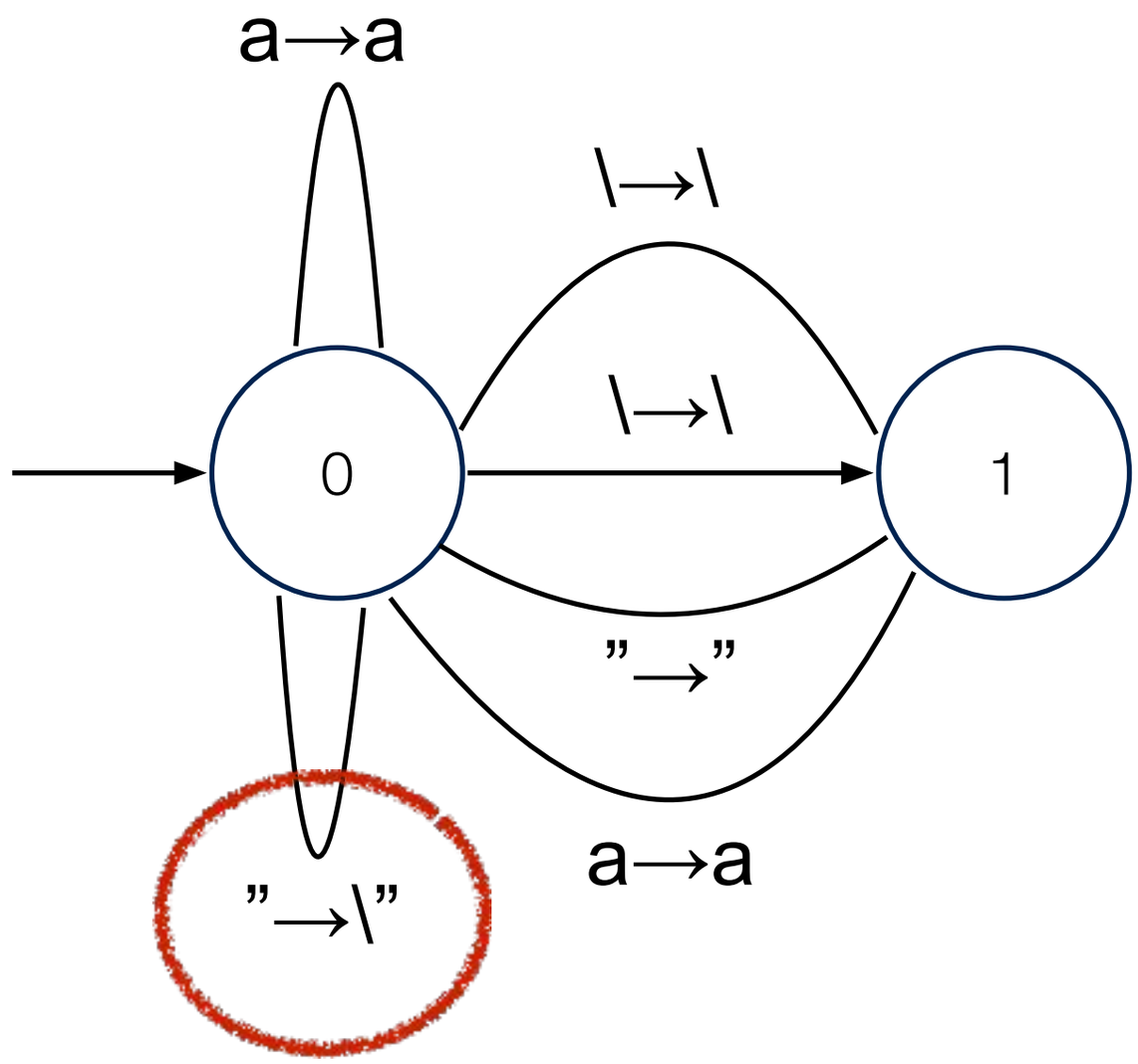
b, B → B

toUpperCase(aBa) → ABA



b, B → B

toUpperCase



escapeQuotes

We automate writing transducers by synthesizing them

Input v_1	*Output*
International Business Machines	*IBM*
Principles Of Programming Languages	*POPL*
International Conference on Software Engineering	*ICSE*

generate abbreviation

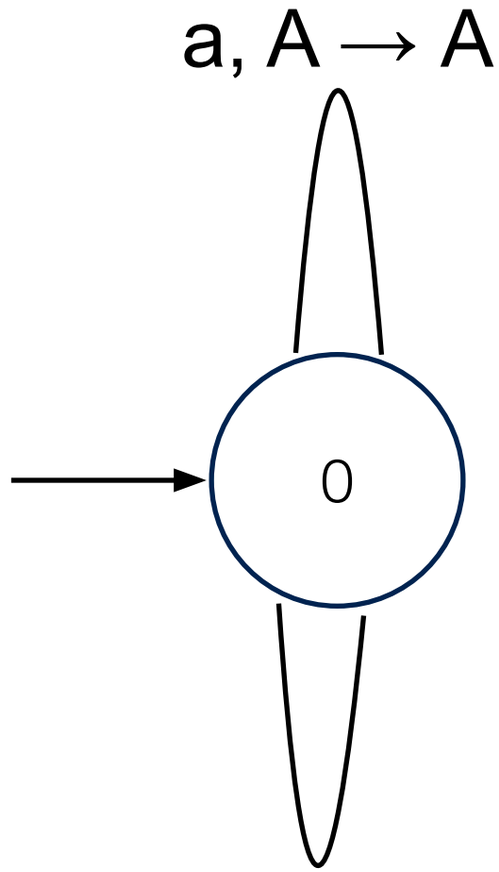
Input v_1	*Output*
BTR KRNL WK CORN 15Z	*15Z*
CAMP DRY DBL NDL 3.6 OZ	*3.6 OZ*
CHORE BOY HD SC SPNG 1 PK	*1 PK*
FRENCH WORCESTERSHIRE 5 Z	*5 Z*
O F TOMATO PASTE 6 OZ	*6 OZ*

extract quantities

[Automating String Processing in Spreadsheets using Input-Output Examples] POPL ‘11

How do we do this?

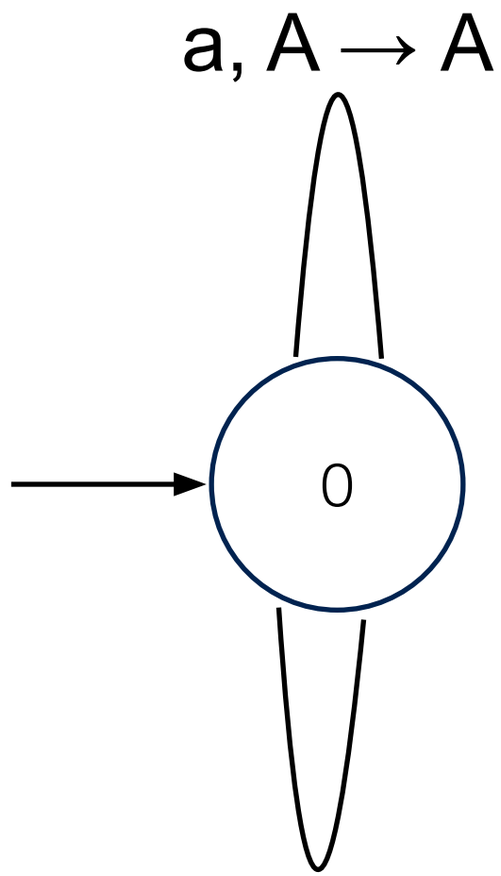
Supporting various specification mechanisms



b, B → B

aBa → ABA

ABA → ABA

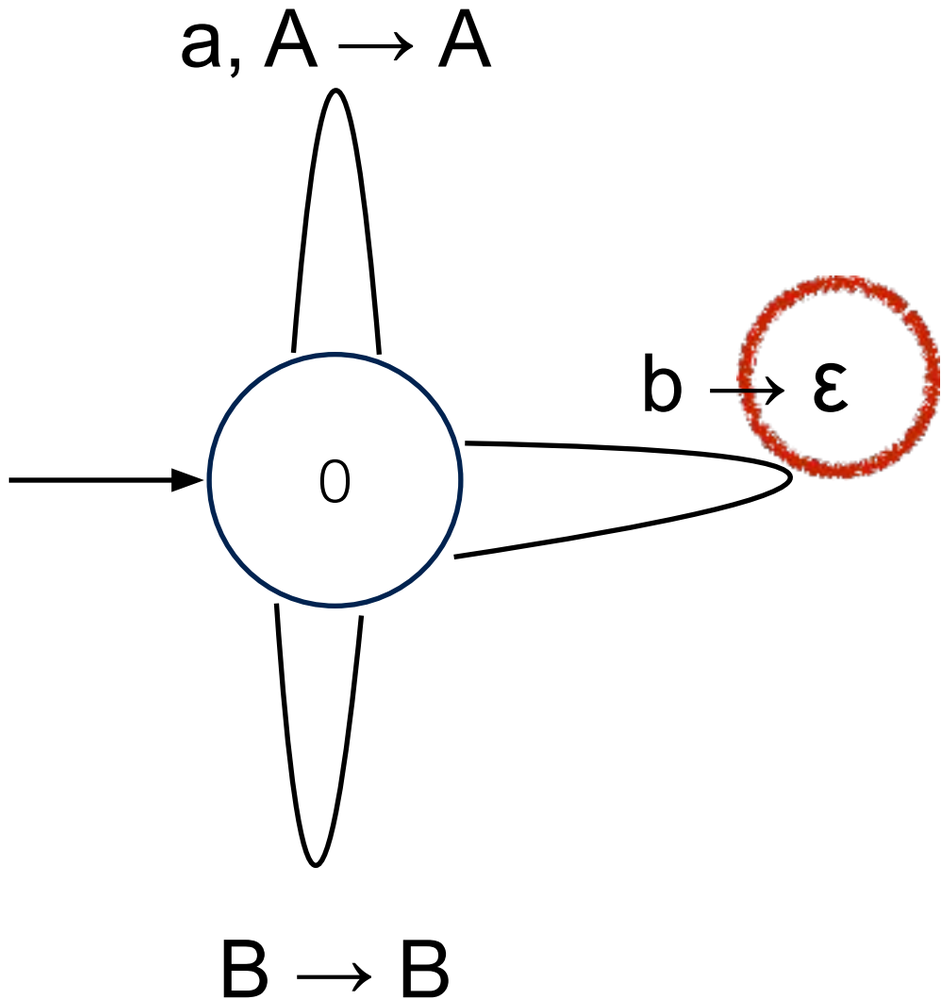


1. Input-Output Examples

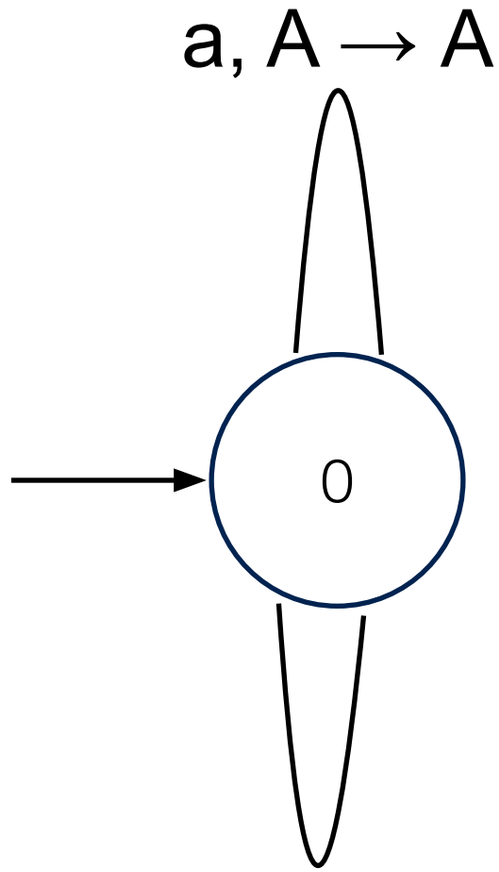
For an example *i* → *o*, we write ***T(i) = o***

aBa → ABA

ABA → ABA

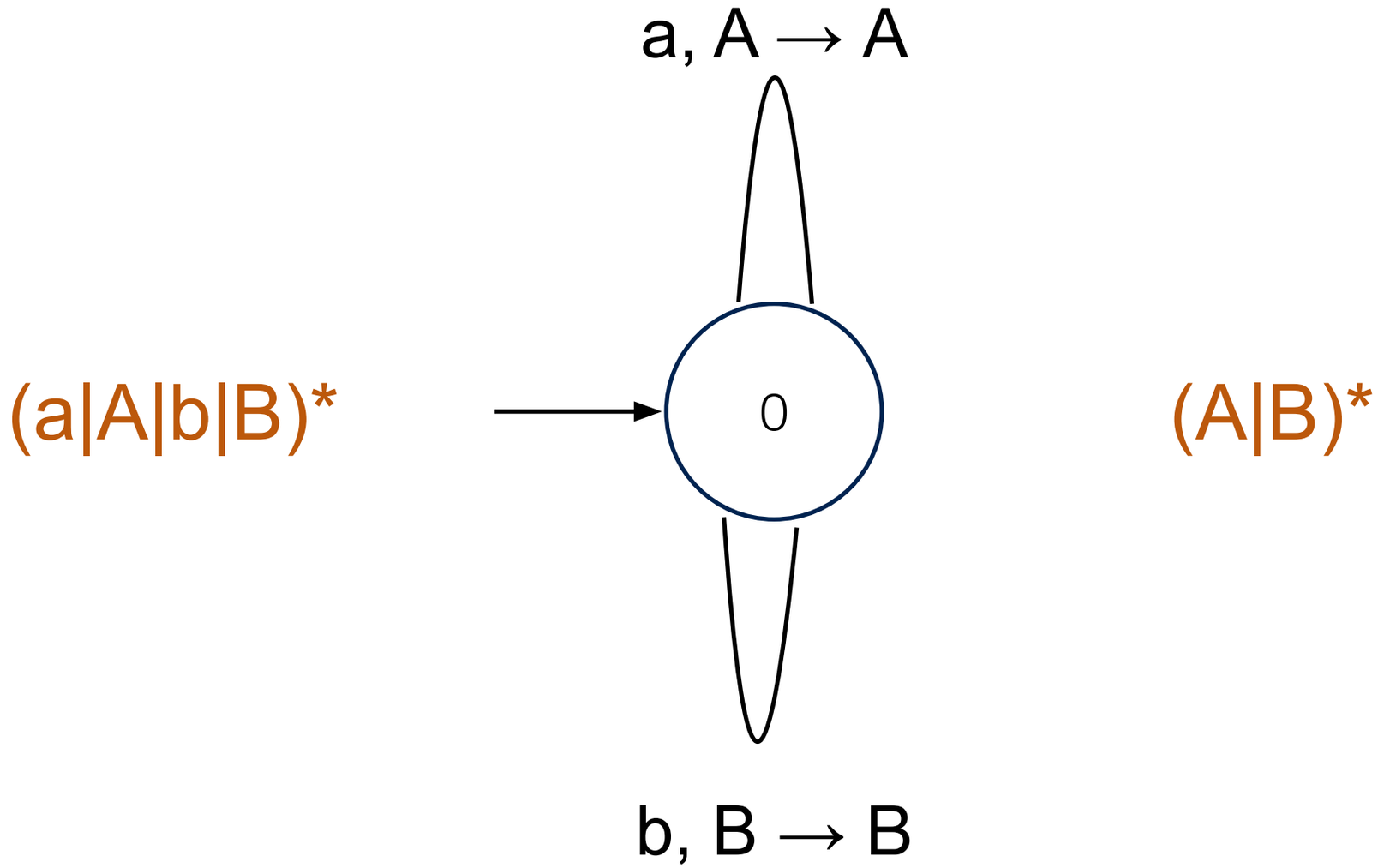


A valid transducer can overfit!



b, B → B

(a|A|b|B)* → (A|B)*



2. Input-Output Types

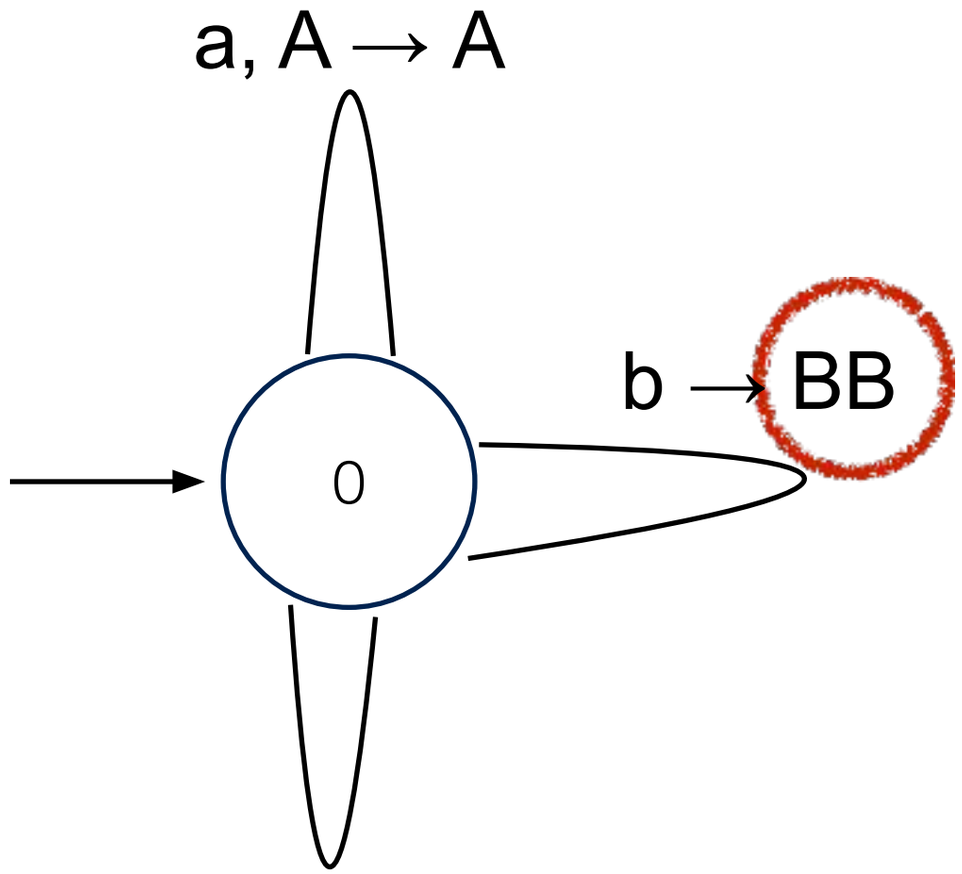
Specify pre and post conditions

For input-output regexes *I*, *O*, we write **{*I*} *T* {*O*}**

aBa → ABA

ABA → ABA

(a|A|b|B)* → (A|B)*



B → B

Still not enough:

Possible to satisfy types in a way that is unintended

$$aBa \rightarrow ABA$$

$$ABA \rightarrow ABA$$

toUpperCase

aBa → ABA

ABA → ABA

toUpperCase

How ‘far apart’ can input-output strings be?

aBa → ABA

ABA → ABA

toUpperCase

3. Input-Output Distance

aBa → ABA

We made 2 edits over a string of length 3

mean edit distance = edits/length = 2/3

Why normalize by length?

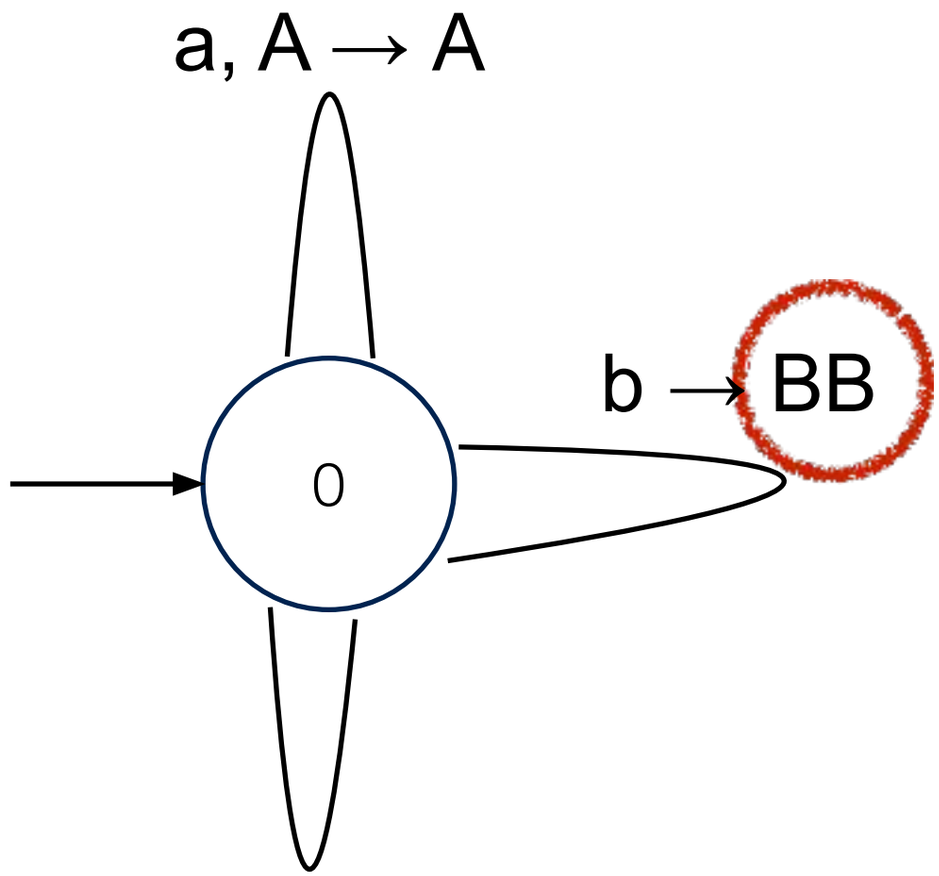
Distance can be bounded even when making an unbounded number of edits

aBa → ABA

For two strings *s, t*, we write ***dist(s, t)*** for their mean edit distance

Transducer *T* satisfies mean edit distance *d* if

$$\forall i \in I.\ dist(i, T(i)) \leq d$$



B → B

To prevent extra characters from being written:

$d = 1/1$

Inputs: regexes *I, O*; examples {<i_1, o_1>,…,<i_n, o_n>}; distance *d*

Output: *T* such that

$$\{ I \}\ T\ \{ O \}$$ **Types**

$$T(i_1) = o_1$$ **Examples**

$$\ldots$$

$$T(i_n) = o_n$$

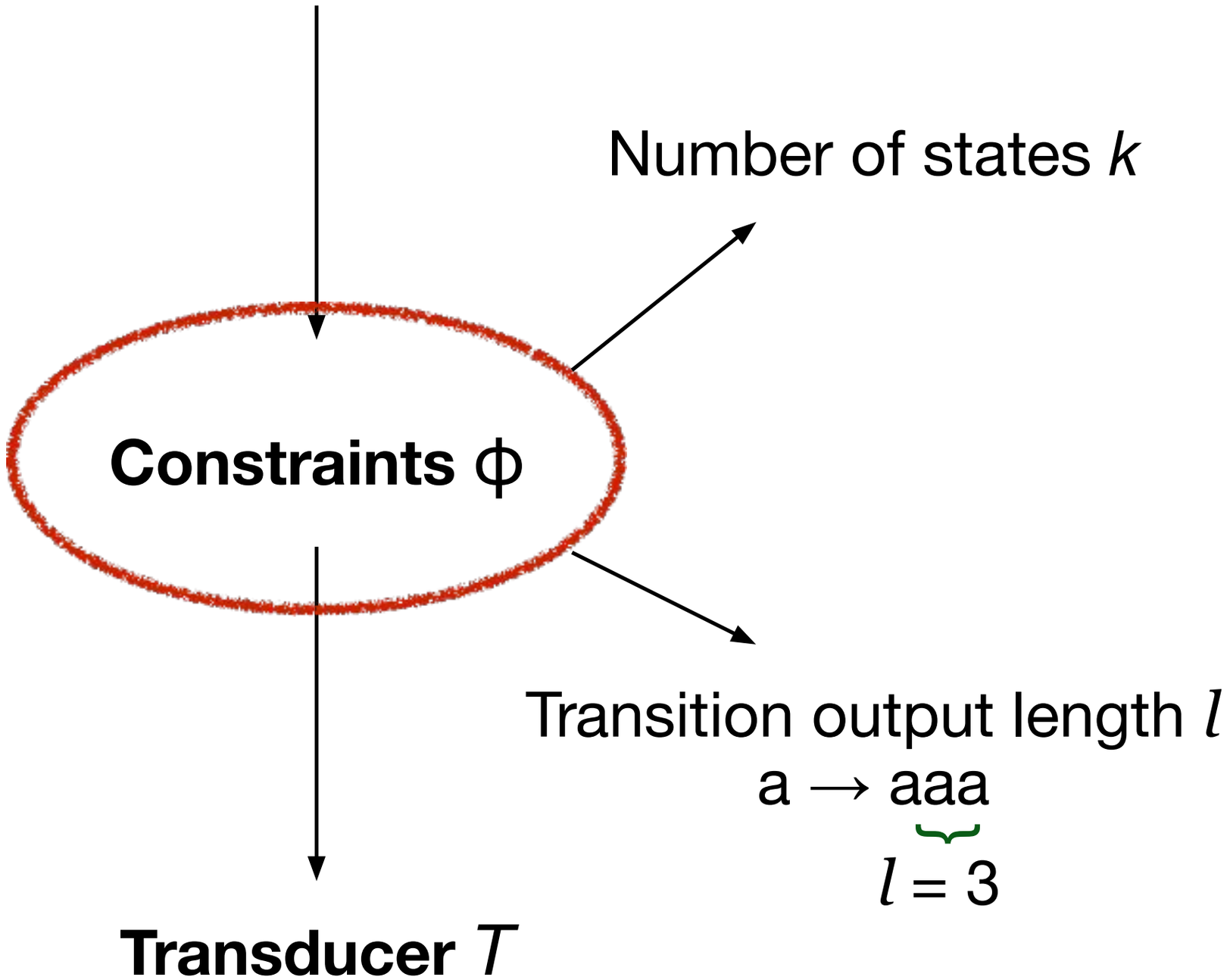
$$\forall i \in I.\ dist(i, T(i)) \leq d$$ **Distance**

Inputs: regexes *I, O*; examples {<i_1, o_1>,…,<i_n, o_n>}; distance *d*

Constraints ϕ

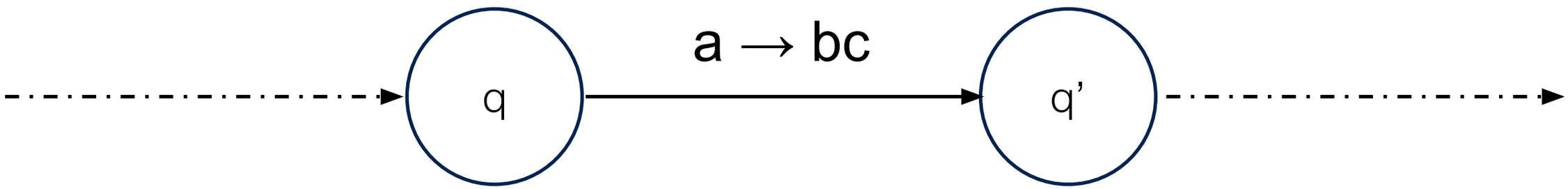
Transducer *T*

Inputs: regexes *I, O*; examples {<i_1, o_1>,…,<i_n, o_n>}; distance *d*



Input-Output Examples

$$x_0 \ldots x_i a \ldots x_n \rightarrow y_0 \ldots y_j bc \ldots y_n$$



config(i) = (j, q)

config(i+1) = (j+2, q')

We have read **i** chars and written **j** chars

For each example:

config: $\mathbb{Z} \rightarrow \mathbb{Z} \times Q_T$

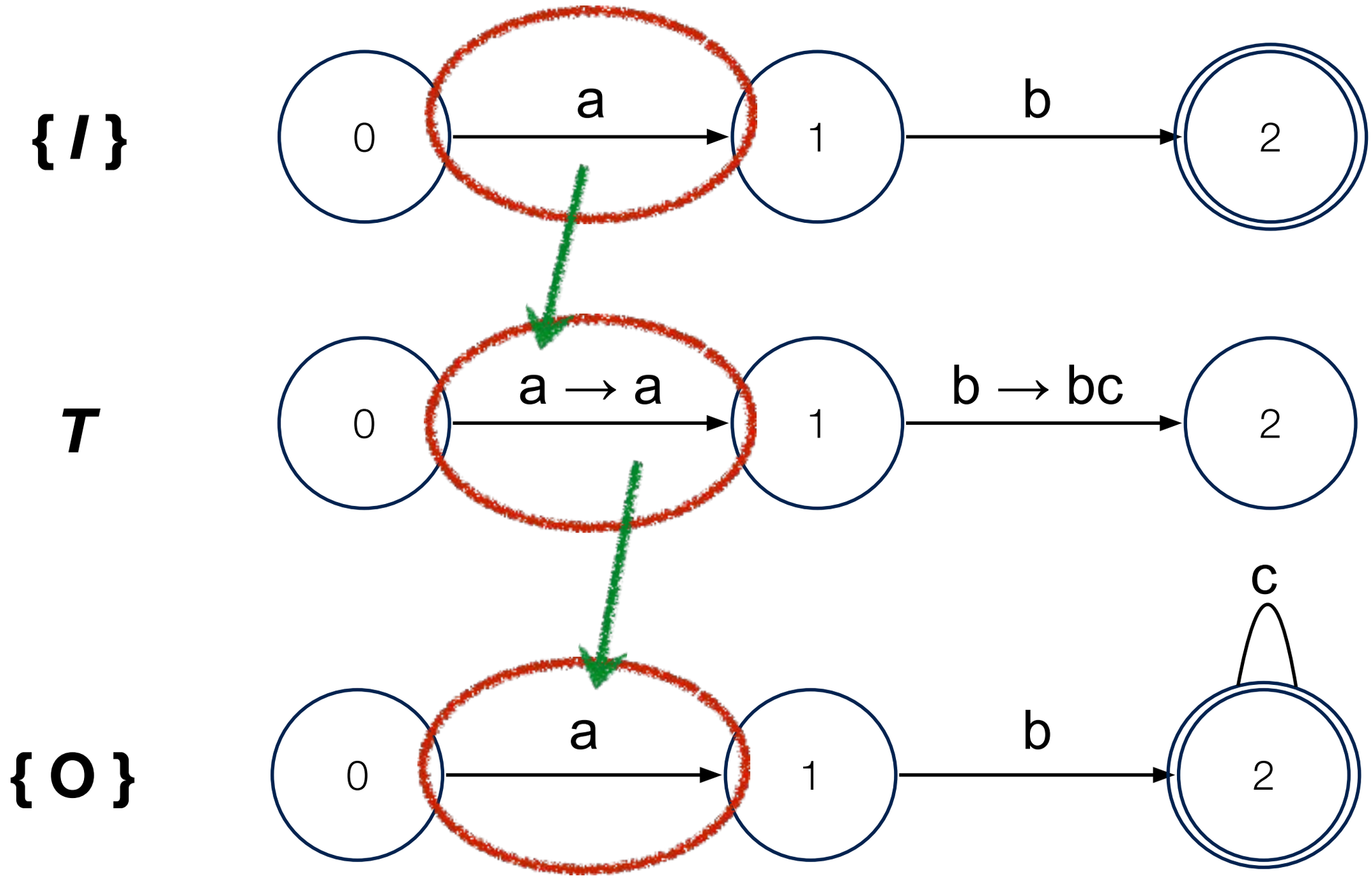
Input Position

Output Position

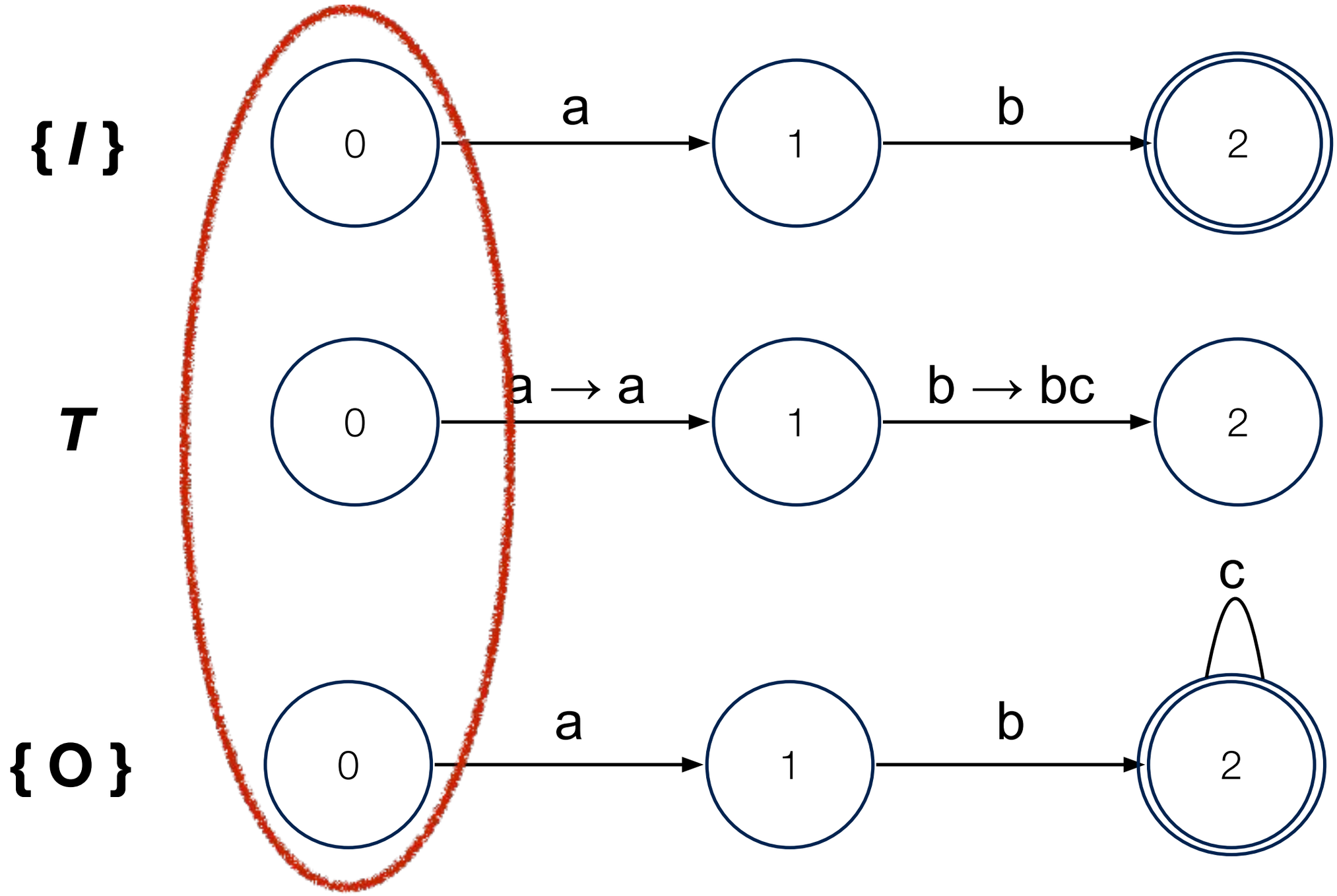
Transducer State

Unknowns!

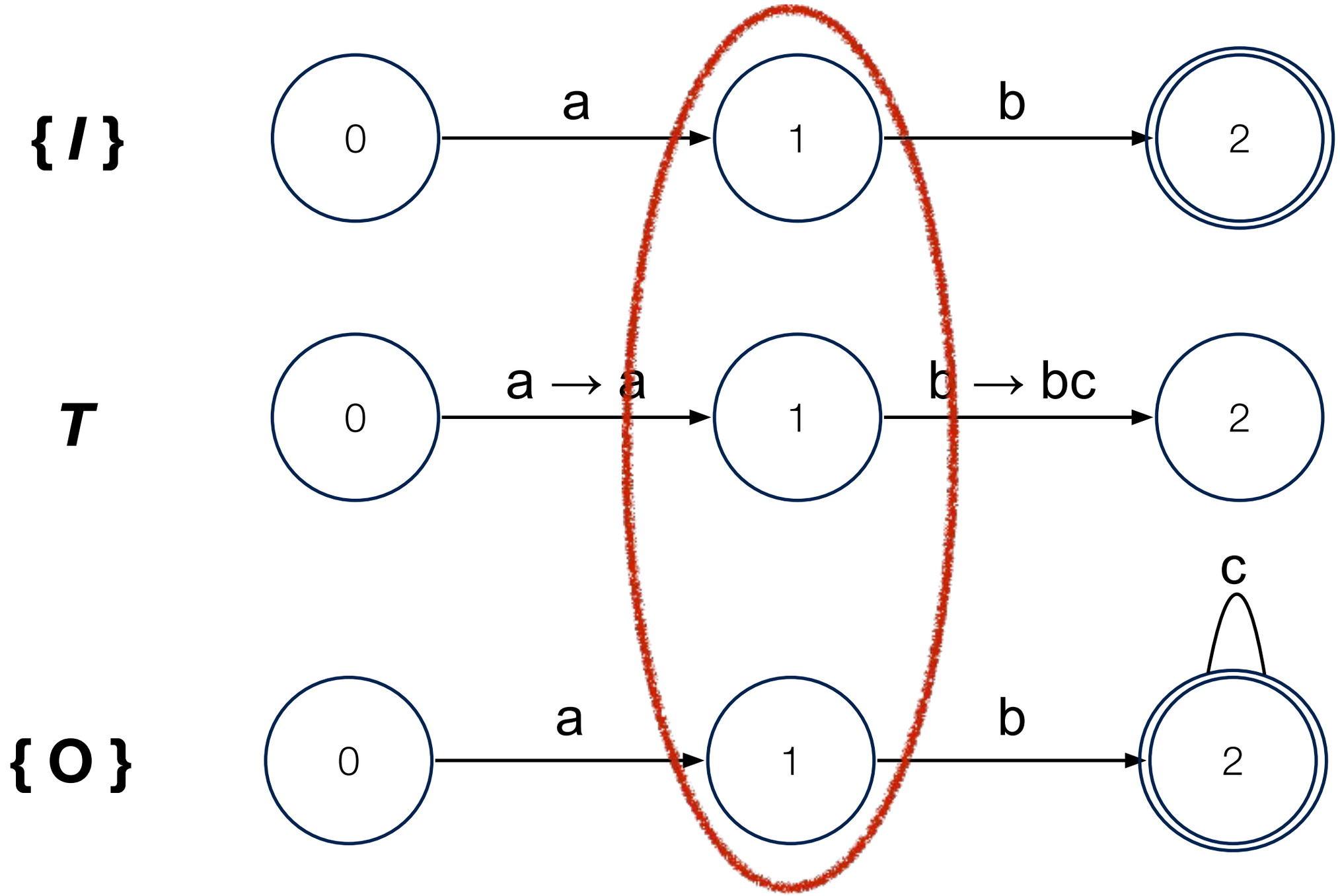
Input-Output Types + Distance



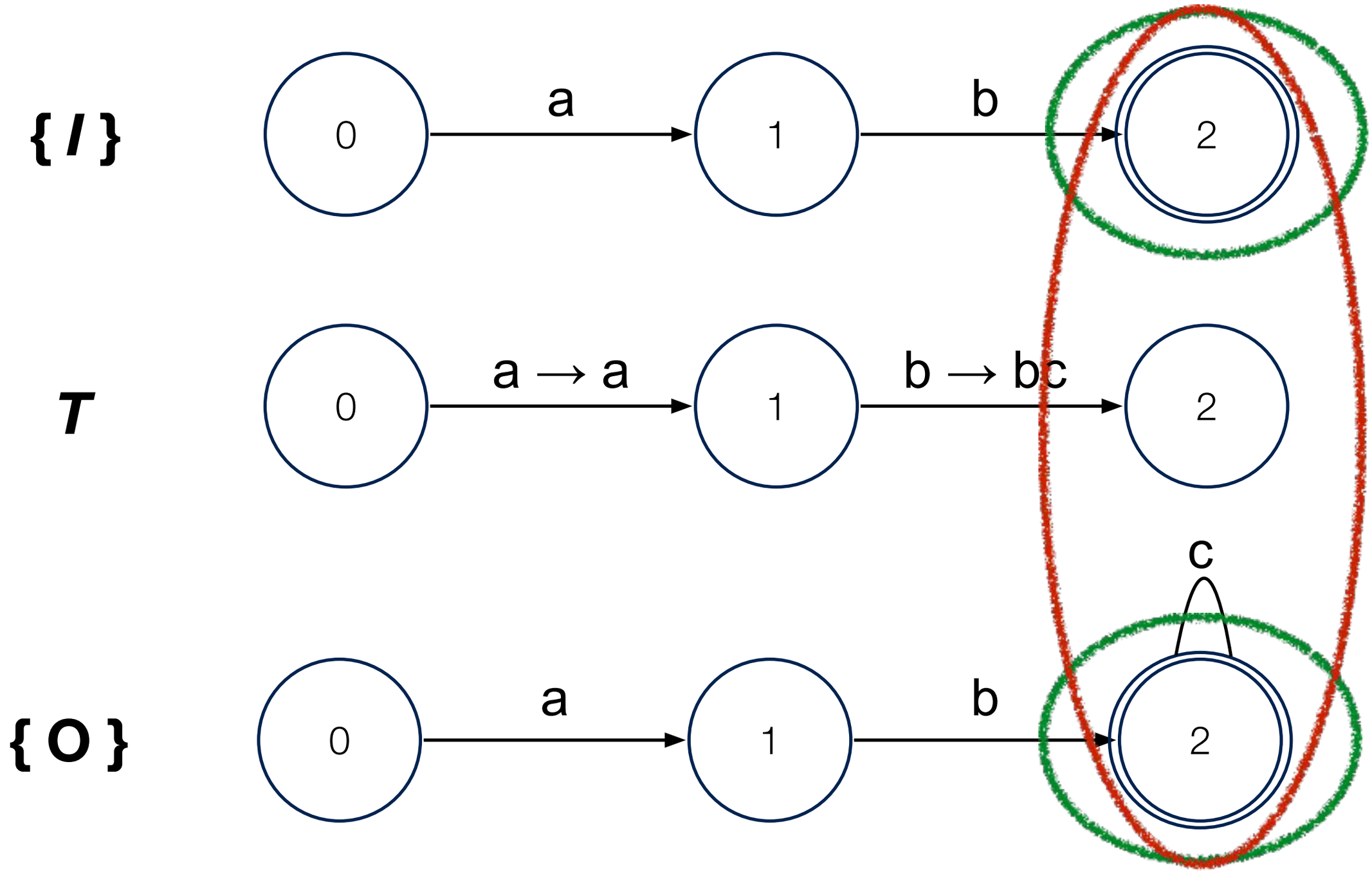
How do we run all 3 automata together?
A simulation relation



$$sim(q^{I}_{init}, q^{T}_{init}, q^{O}_{init})$$

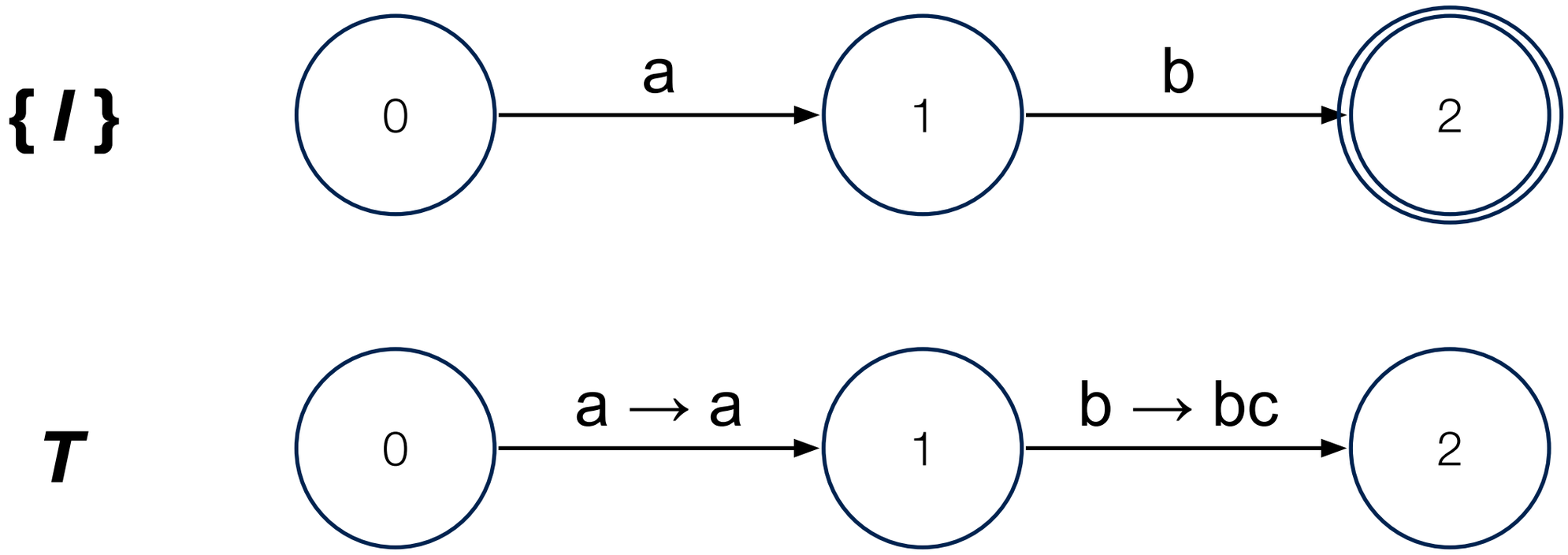


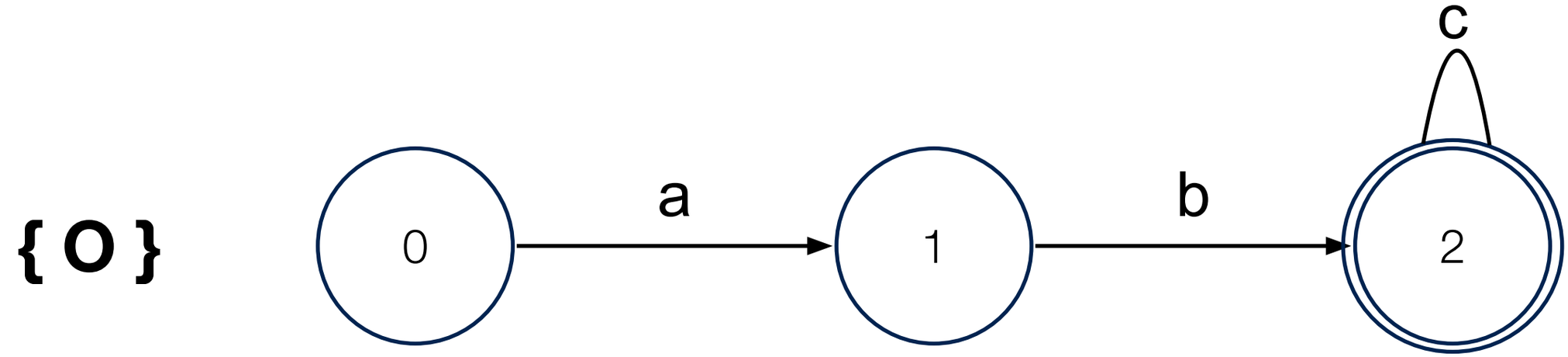
$$sim(q^{I}_{1}, q^{T}_{1}, q^{O}_{1})$$



$sim(q^I, q^T, q^O)$

If q^I is final, q^O should also be final

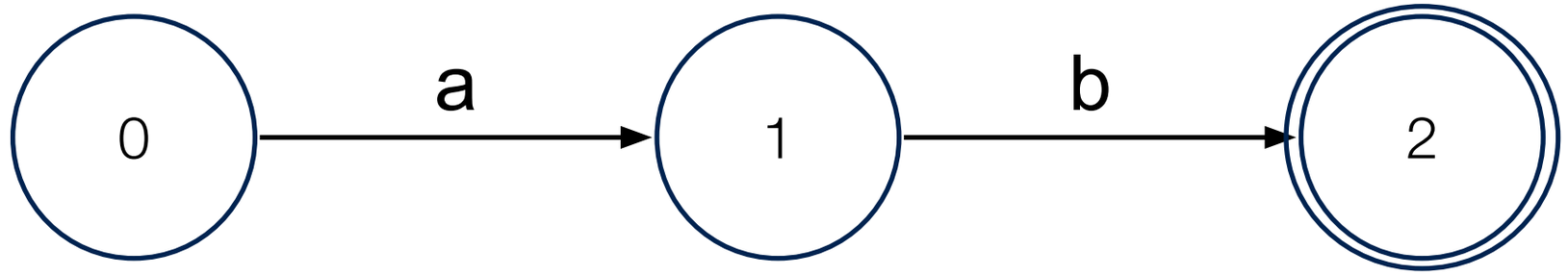




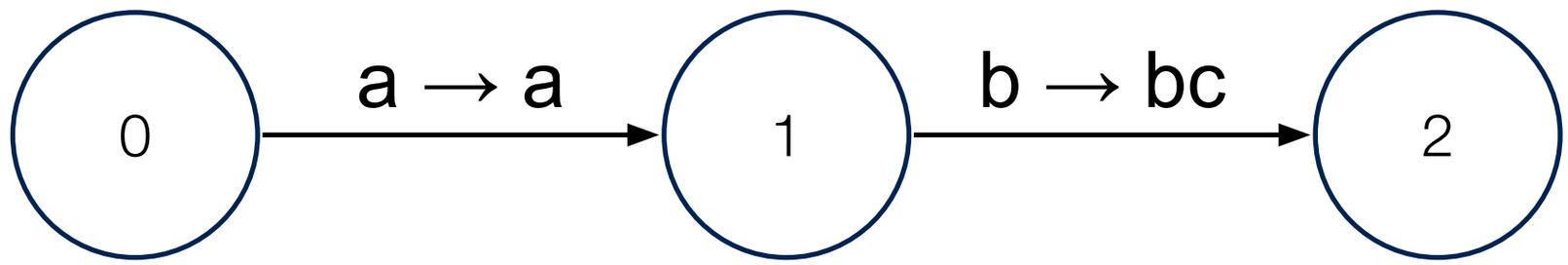
Now what about distance?

Associate a budget value with each point in the simulation

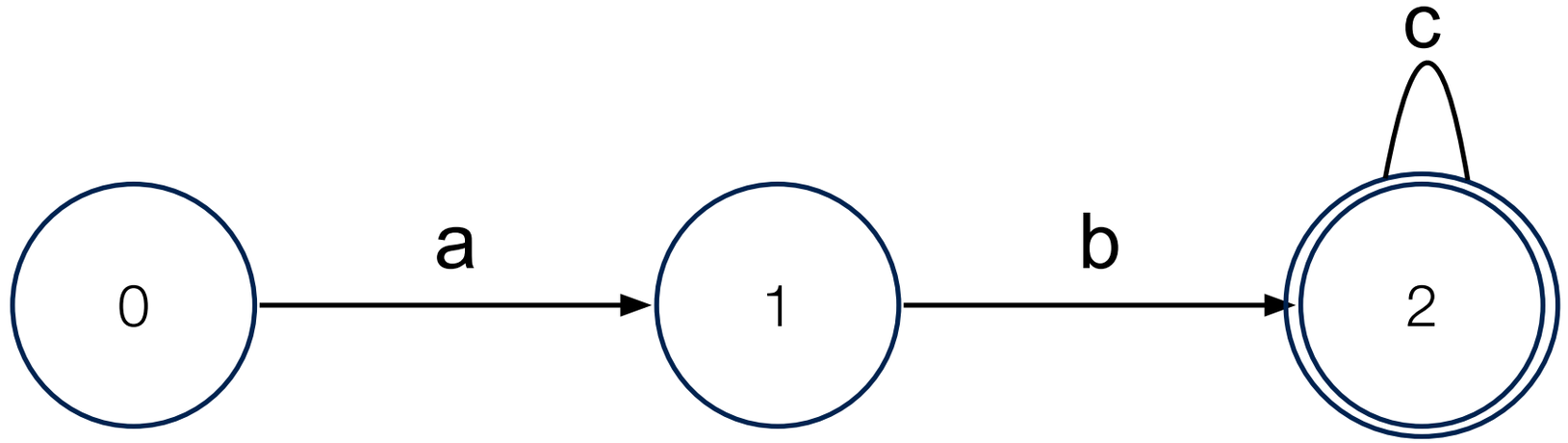
{ I }



T



{ O }



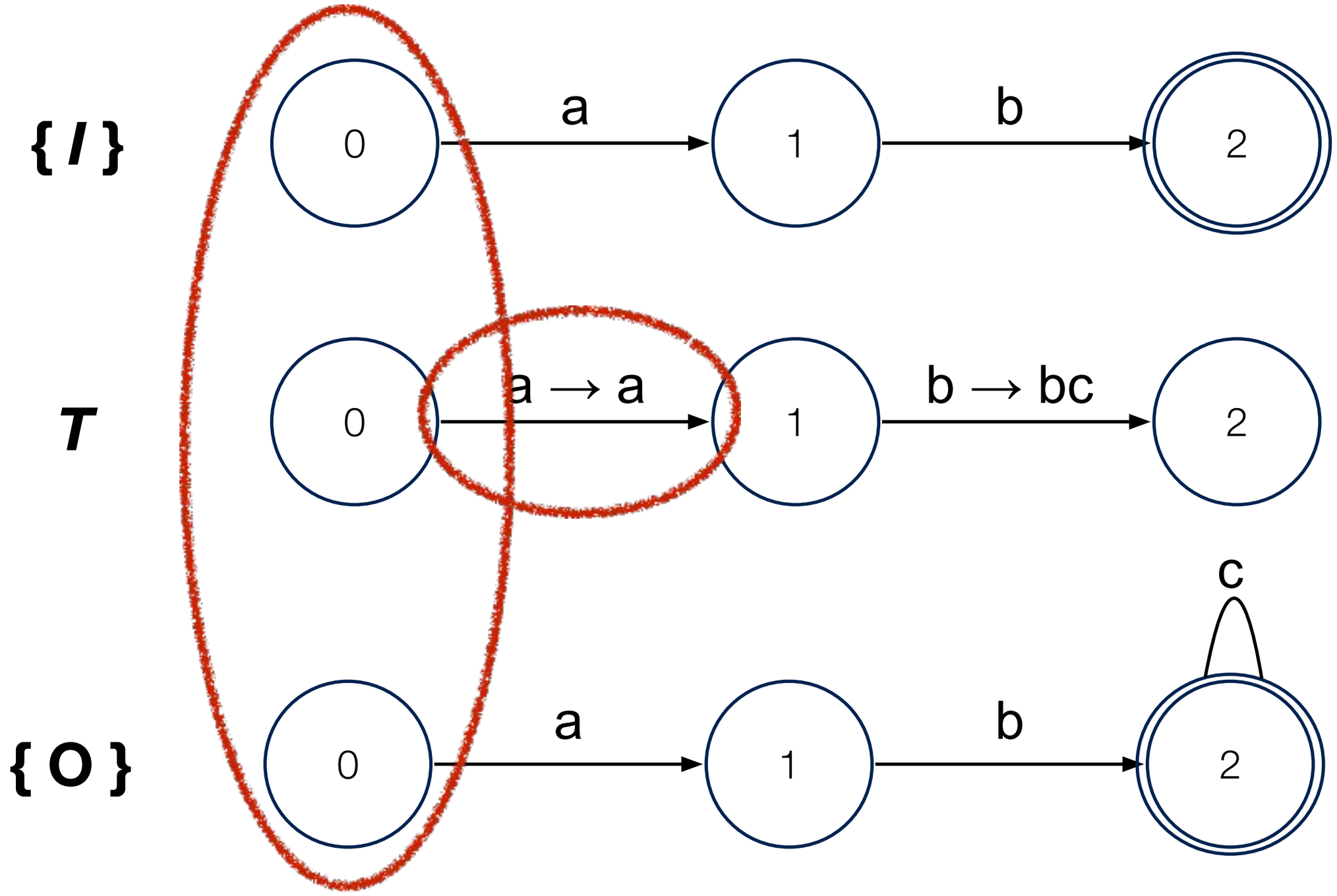
d =

1/2

Goal: $\forall i \in I.\ dist(i, T(i)) \leq 1/2$

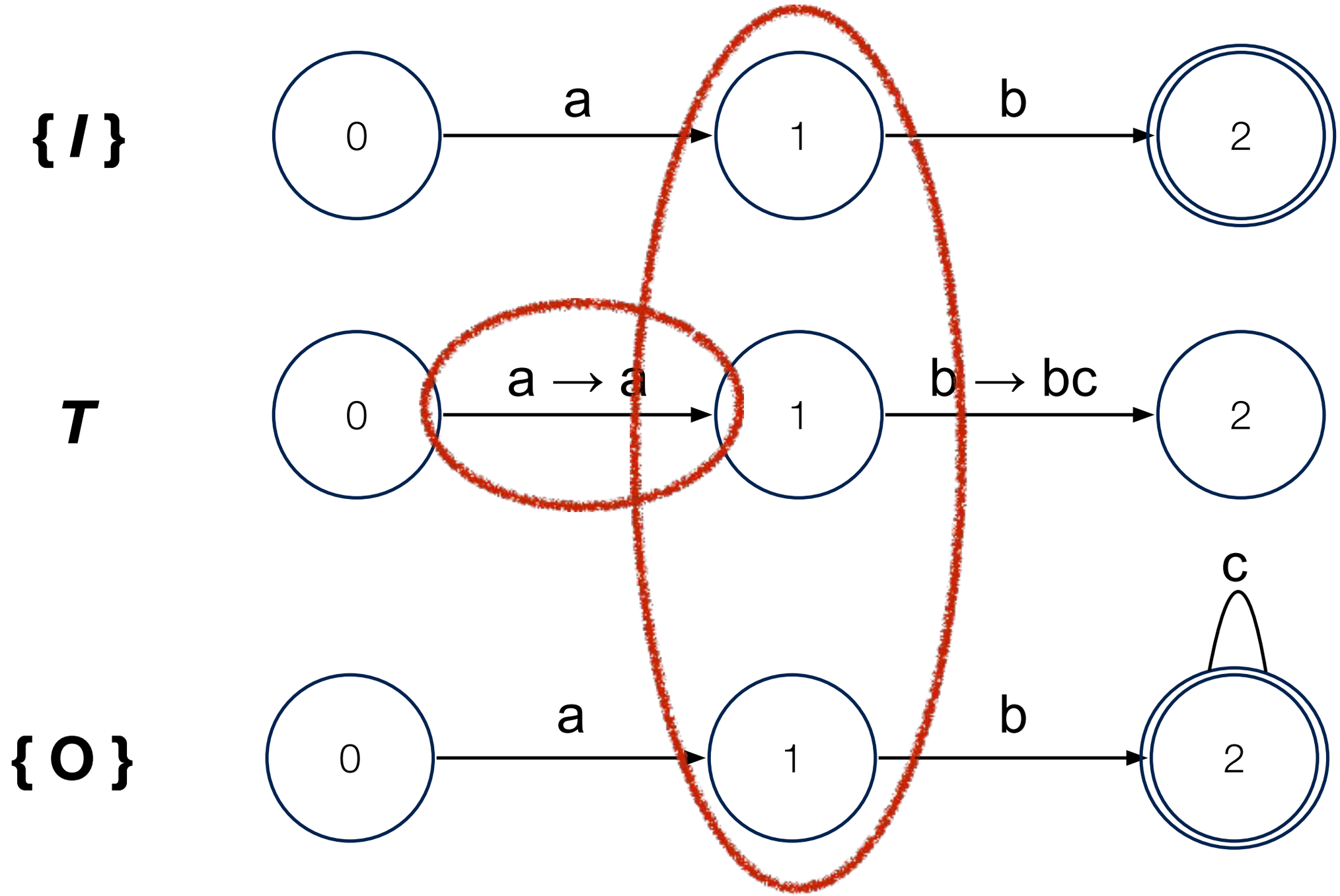
Want (budget >= 0) to not be in a deficit

Goal: $\forall i \in I.\ \mathrm{dist}(i, T(i)) \leq 1/2$

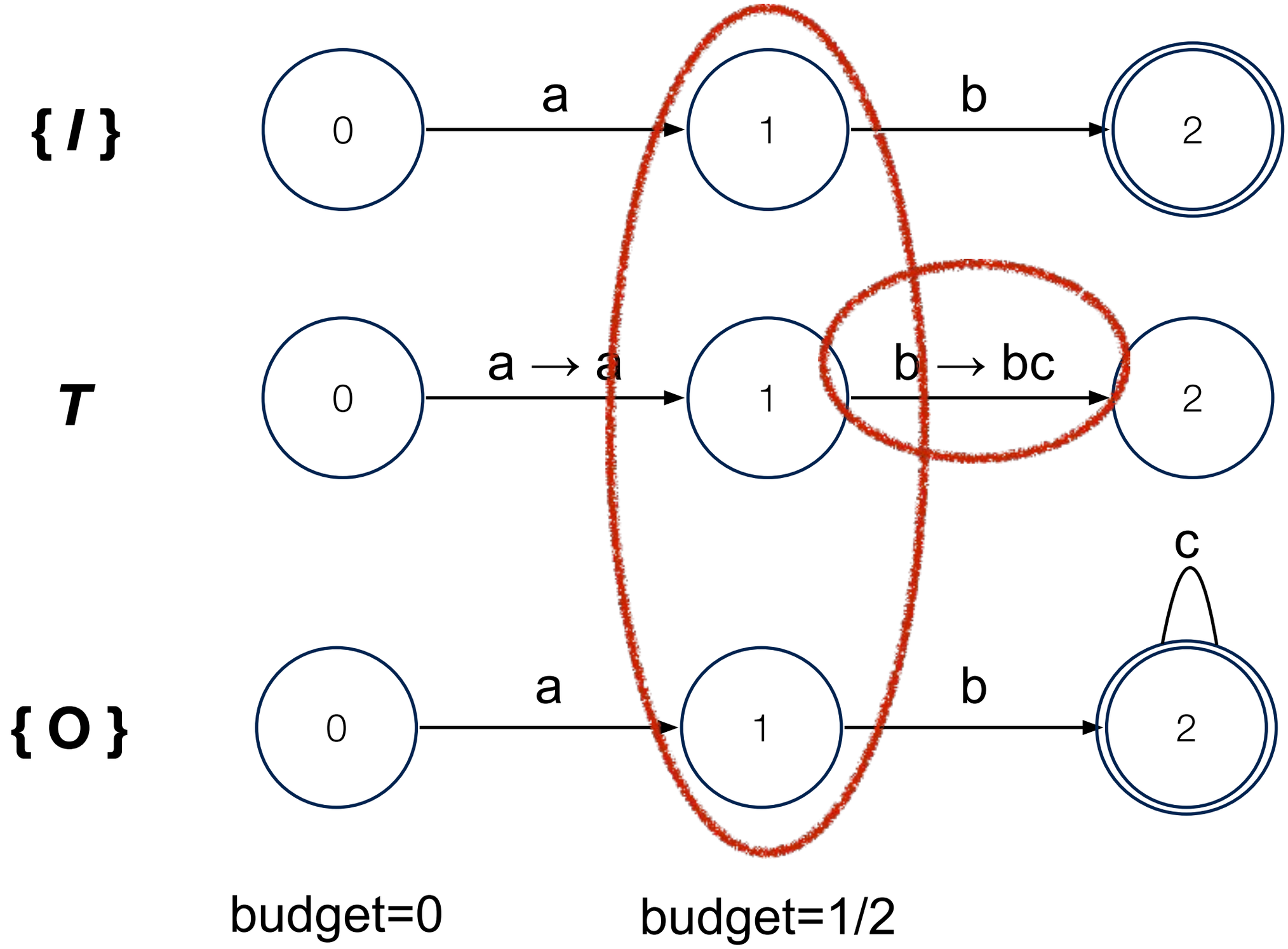


budget=0

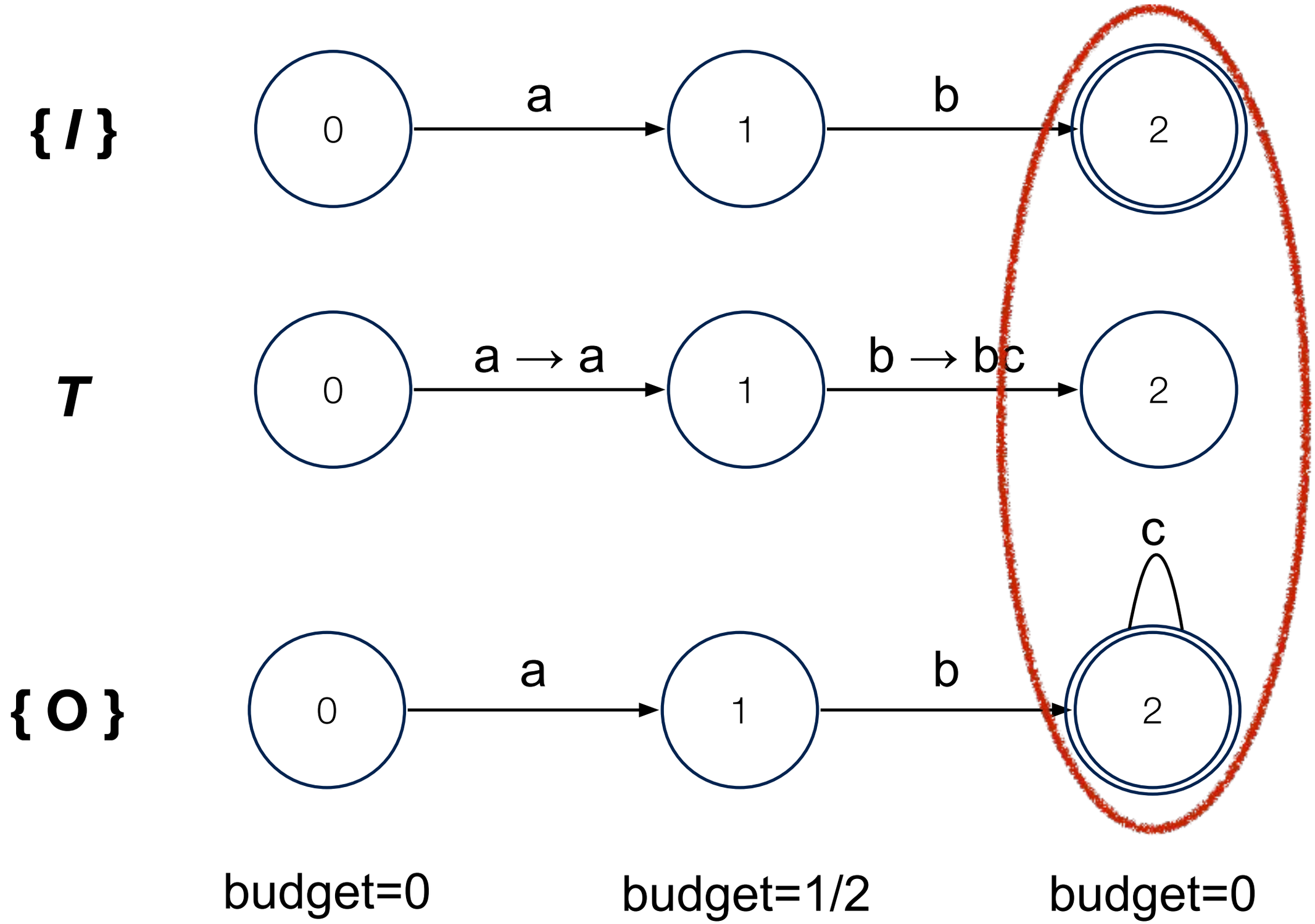
Goal: $\forall i \in I.\ \mathrm{dist}(i, T(i)) \leq 1/2$



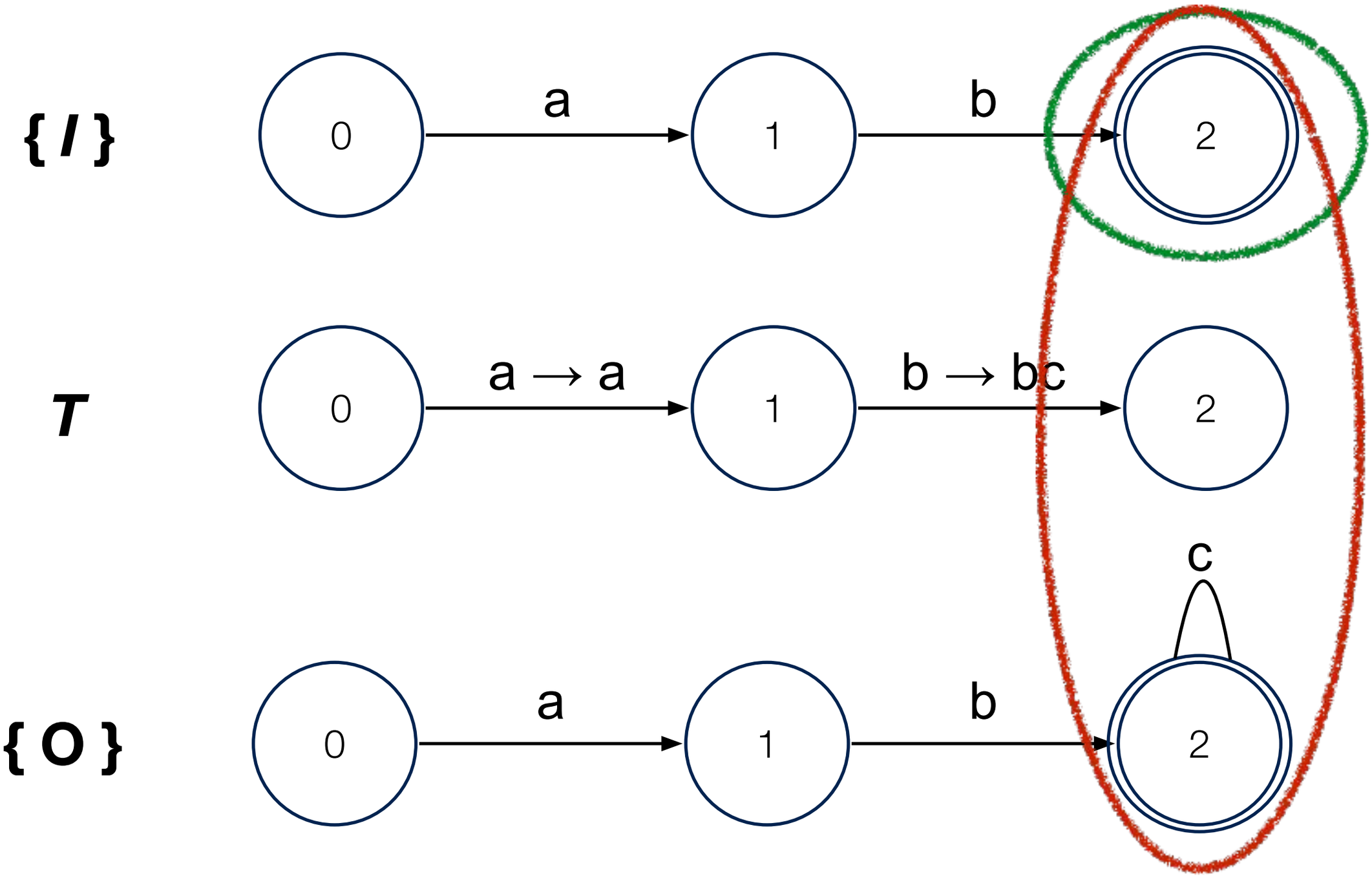
Goal: $\forall i \in I.\ \mathrm{dist}(i, T(i)) \leq 1/2$



Goal: $\forall i \in I.\ \mathrm{dist}(i, T(i)) \leq 1/2$

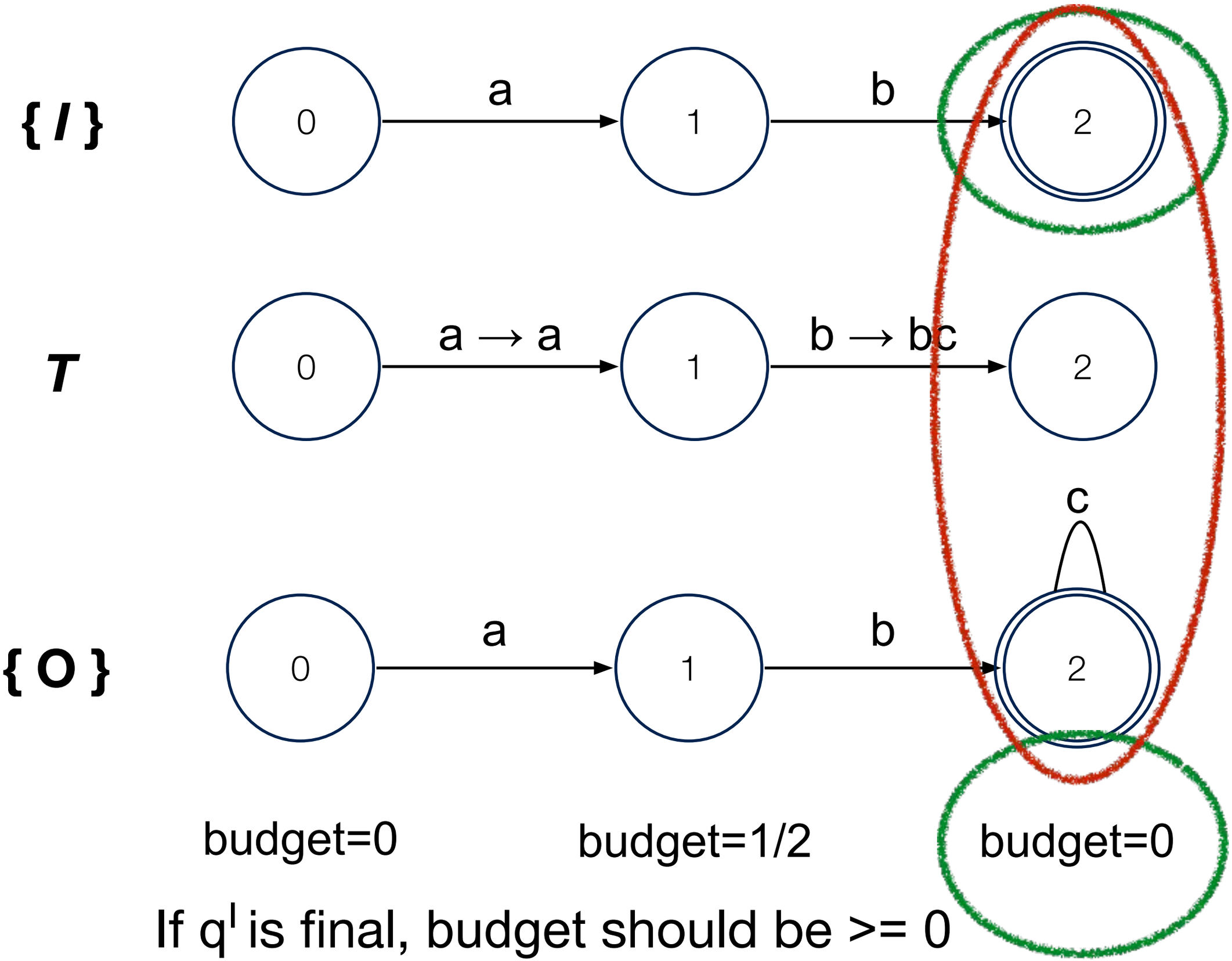


Goal: $\forall i \in I.\ dist(i, T(i)) \leq 1/2$



If q^I is final, budget should be >= 0

Goal: $\forall i \in I.\ dist(i, T(i)) \leq 1/2$



If q^I is final, budget should be >= 0

Constraints use uninterpreted functions:

$$\text{sim: } Q_I \times Q_T \times Q_O \rightarrow \{0, 1\}$$

$$\text{budget: } Q_I \times Q_T \times Q_O \rightarrow \mathbb{Z}$$

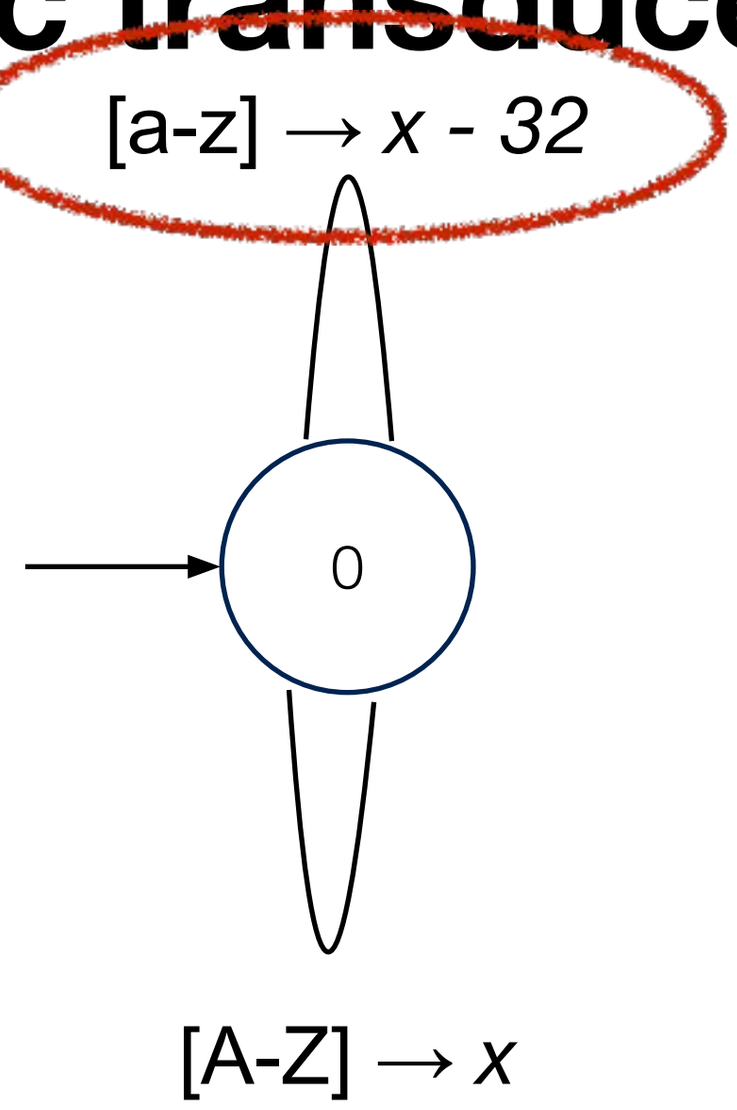
budget computation is sound but incomplete:

We operate at transition level instead of string level

Also in the paper:

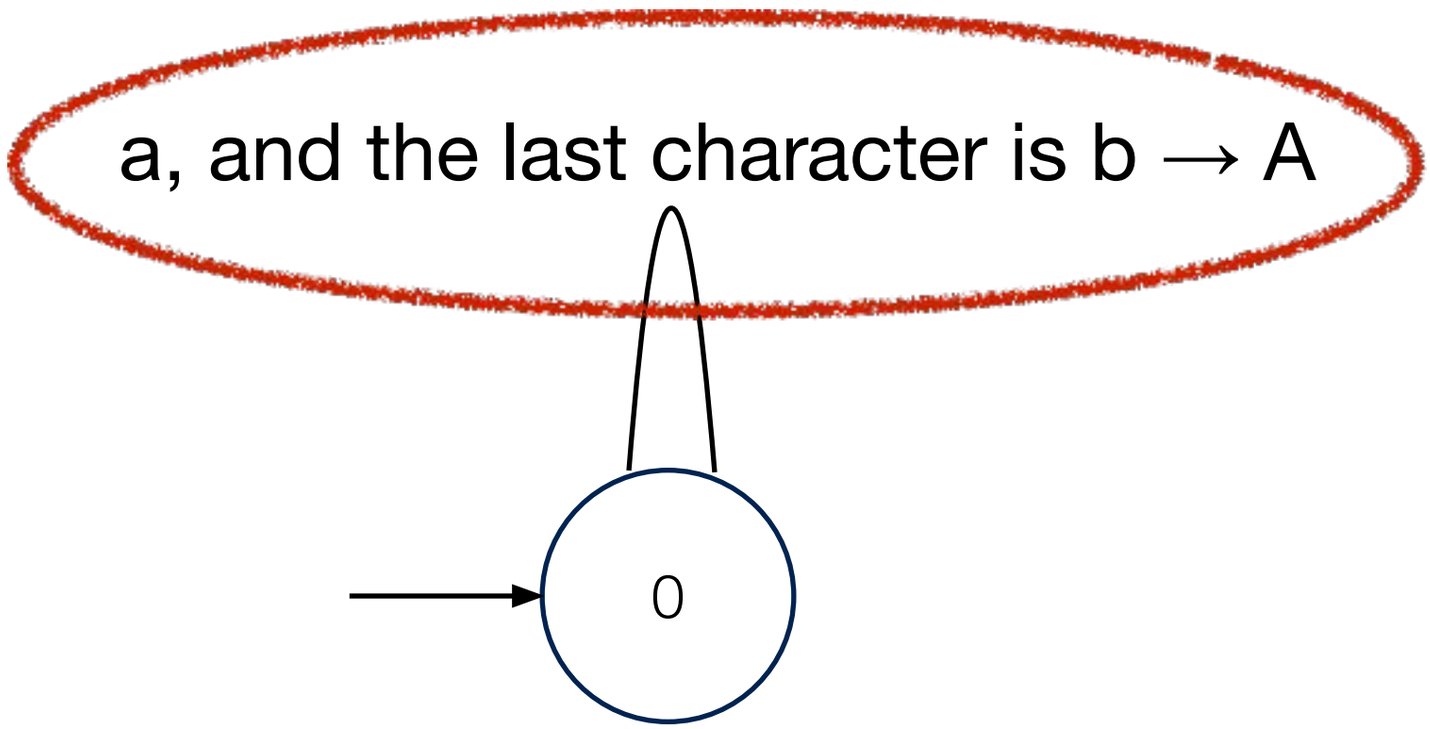
Synthesizing symbolic transducers

symbolic toUpperCase



Also in the paper:

Synthesizing transducers with lookahead (non-determinism)



Lookahead: Know what is going to come later

Evaluation

Implemented in a tool: Astra

Benchmarks

What can we evaluate on?

Optician: Tool for synthesizing bidirectional transformations

Also uses IO types+examples

[Synthesizing Symmetric Lenses] ICFP‘19

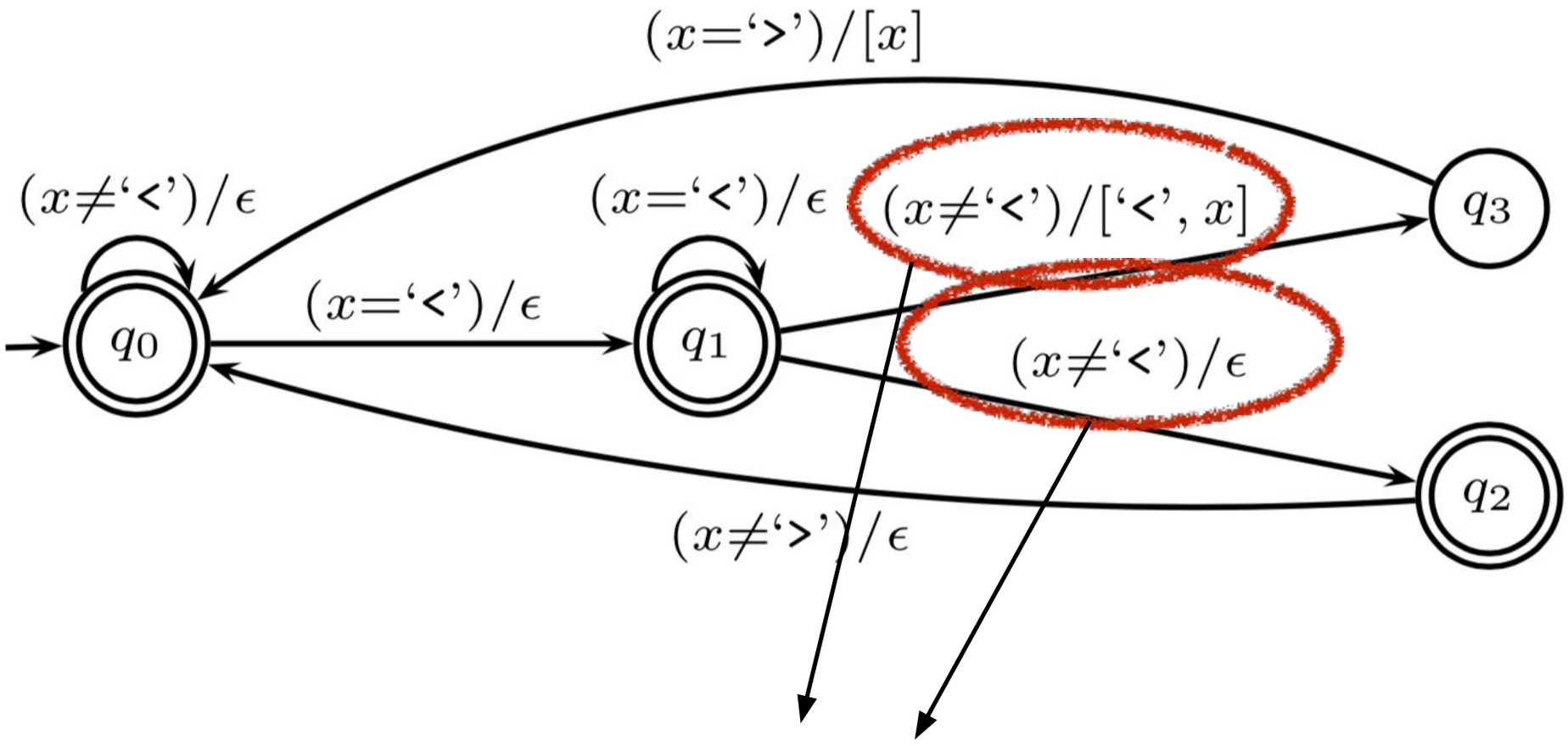
Benchmarks:

1. Optician's benchmark suite (data transformation tasks)

2. Examples from work on symbolic transducers

The ‘getTags’ transducer

[Symbolic Finite State Transducers: Algorithms and Applications] POPL ‘12



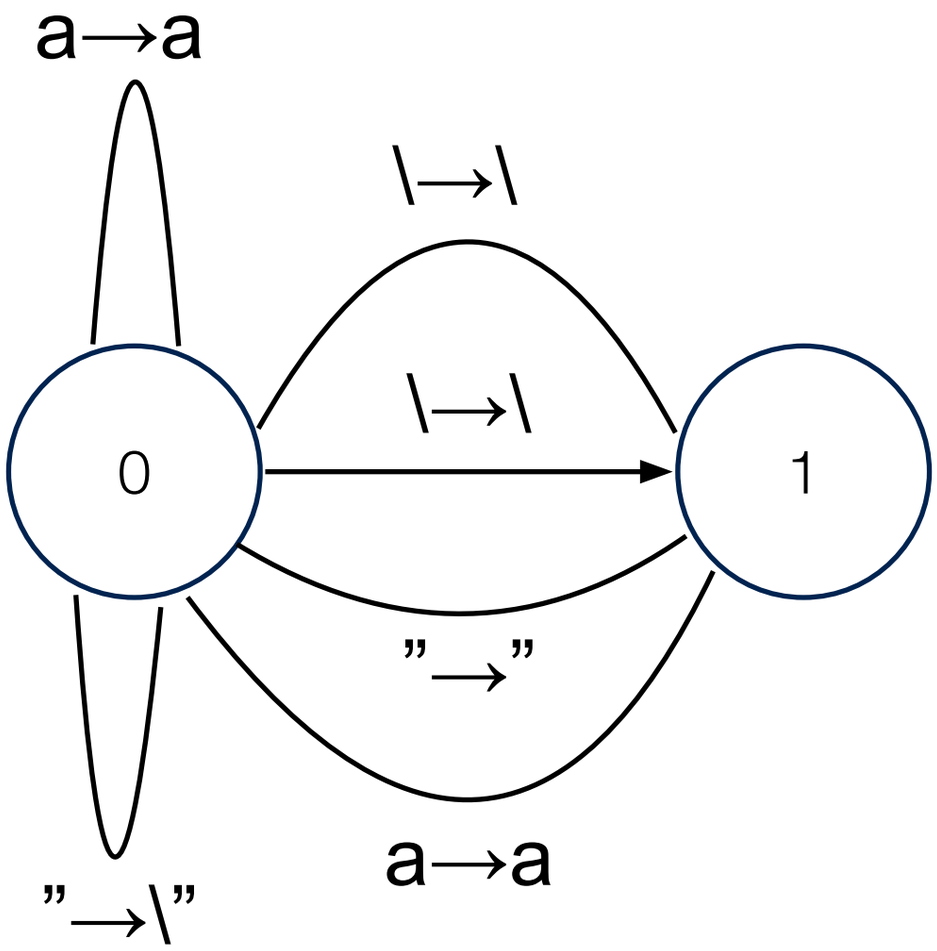
Same input predicate (nondeterministic)

Benchmark	ASTRA (s)	Optician (s)
extrAcronym	0.11	0.05
extrAcronym2	0.42	—
extrNum	0.93	0.05
extrQuant	0.19	0.09
normalizeSpaces	0.46	16.64
extrOdds	15.87	0.12
capProb	0.05	0.05
removeLast	0.21	0.15
sourceToViews	50.92	0.06
normalizeNamePos	✗	0.05
titleConverter	✗	0.07
bibtextToReadable	✗	0.64

Benchmark	ASTRA (s)	Optician (s)
extrAcronym	0.11	0.05
extrAcronym2	0.42	—
extrNum	0.93	0.05
extrQuant	0.19	0.09
normalizeSpaces	0.46	16.64
extrOdds	15.87	0.12
capProb	0.05	0.05
removeLast	0.21	0.15
sourceToViews	50.92	0.06
normalizeNamePos	✗	0.05
titleConverter	✗	0.07
bibtextToReadable	✗	0.64

}

>10 states in the input automaton



escapeQuotes: replace every ” with \”

Optician cannot synthesize this

Working with unstructured data

regexes *I*, *O* for Input-Output Types

Automata

Back to regex

[A-Z][a-z]*

((([A-Z] . [a-z]) . [a-z]*) | [A-Z])

Benchmark	ASTRA (s)	Optician (s)	Optician-re (s)
extrAcronym	0.11	0.05	✗
extrAcronym2	0.42	—	—
extrNum	0.93	0.05	0.07
extrQuant	0.19	0.09	✗
normalizeSpaces	0.46	16.64	✗
extrOdds	15.87	0.12	✗
capProb	0.05	0.05	✗
removeLast	0.21	0.15	0.07
sourceToViews	50.92	0.06	✗
normalizeNamePos	✗	0.05	0.10
titleConverter	✗	0.07	✗
bibtextToReadable	✗	0.64	0.15

We find T such that:

$\{I\}\ T\ \{O\}$ **Types**

$T(i_1) = o_1$

...

$T(i_n) = o_n$ **Examples**

$\forall i \in I.\ dist(i, T(i)) \leq d$ **Distance**

Summary

Astra synthesizes:

- Finite Transducers
- Symbolic Transducers
- Transducers with Lookahead

Future work:

- Scalability
- Invariants for string manipulating programs