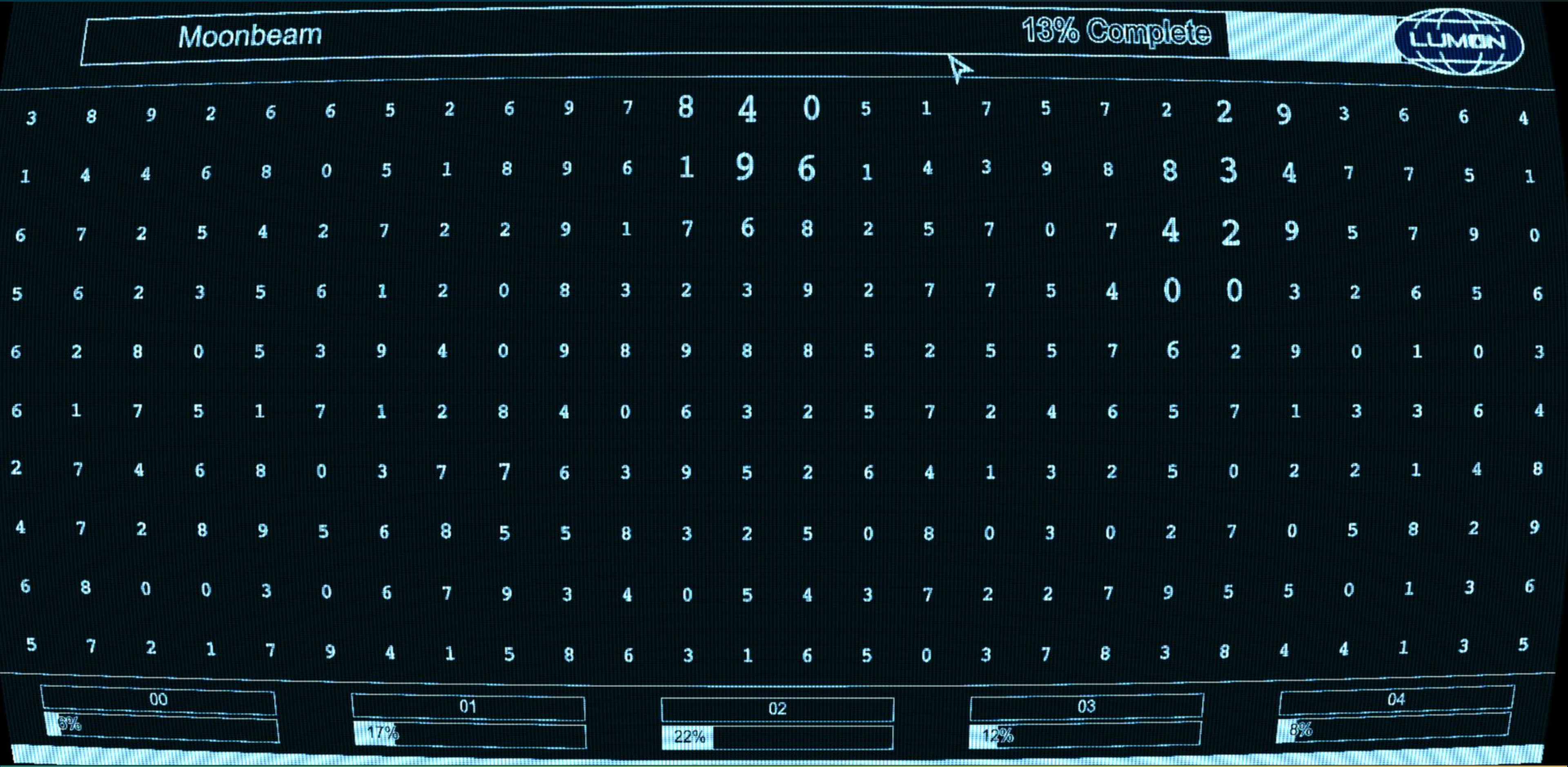
Cyolo The Access Company For The Digital Enterprise

The Ultimate Link Rewriting Hack

whoami

- Amit Yahav, 29 years old.
- Working at Cyolo for 1.5 years in the Bitwise team.
- Excited about low level concepts - OSs, DBs, etc.
- Guitar Player.

What do we do at Cyolo

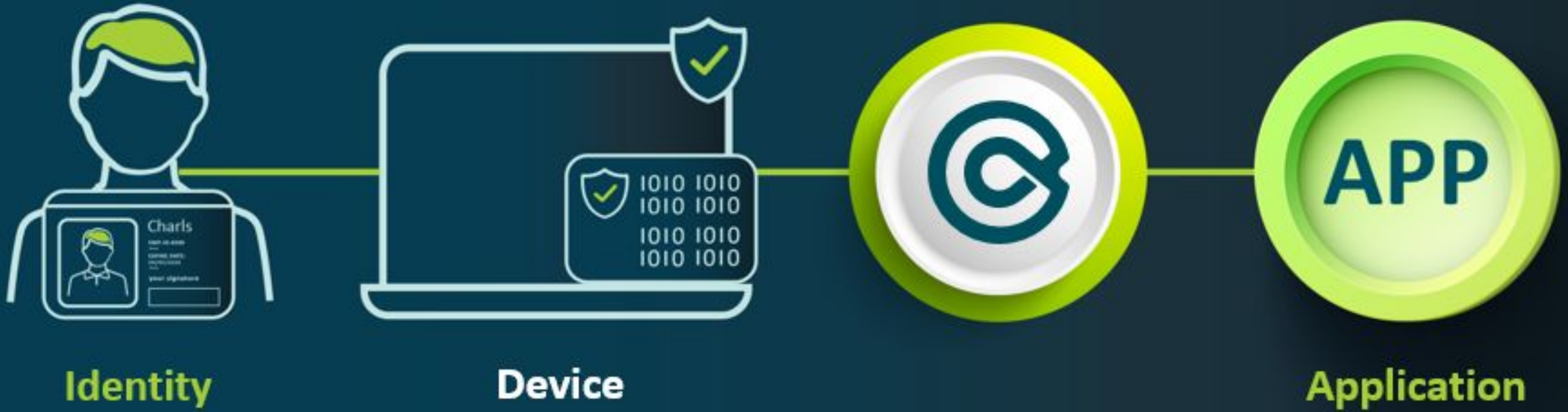


```
3 8 9 2 6 6 5 2 6 9 7 8 4 0 5 1 7 5 7 2 2 9 3 6 6 4
1 4 4 6 8 0 5 1 8 9 6 1 9 6 1 4 3 9 8 8 3 4 7 7 5 1
6 7 2 5 4 2 7 2 2 9 1 7 6 8 2 5 7 0 7 4 2 9 5 7 9 0
5 6 2 3 5 6 1 2 0 8 3 2 3 9 2 7 7 5 4 0 0 3 2 6 5 6
6 2 8 0 5 3 9 4 0 9 8 9 8 8 5 2 5 5 7 6 2 9 0 1 0 3
6 1 7 5 1 7 1 2 8 4 0 6 3 2 5 7 2 4 6 5 7 1 3 3 6 4
2 7 4 6 8 0 3 7 7 6 3 9 5 2 6 4 1 3 2 5 0 2 2 1 4 8
4 7 2 8 9 5 6 8 5 5 8 3 2 5 0 8 0 3 0 2 7 0 5 8 2 9
6 8 0 0 3 0 6 7 9 3 4 0 5 4 3 7 2 2 7 9 5 5 0 1 3 6
5 7 2 1 7 9 4 1 5 8 6 3 1 6 5 0 3 7 8 3 8 4 4 1 3 5
```

00	01	02	03	04
6%	17%	22%	12%	8%

We connect Verified Identities To Applications With Continuous Authorization

*Instead of users to networks



Starting From Your Biggest Risks

All of the customer’s traffic is proxied via the IDAC

All of the customer’s traffic is proxied via the IDAC

– SSH , RDP proxy:

All of the customer’s traffic is proxied via the IDAC

– SSH , RDP proxy:

 - Recording

All of the customer’s traffic is proxied via the IDAC

– SSH , RDP proxy:

- Recording
- Indexing

All of the customer's traffic is proxied via the IDAC

– SSH , RDP proxy:

  - Recording
  - Indexing

- MySQL, PSQL proxy

All of the customer’s traffic is proxied via the IDAC

– SSH , RDP proxy:

  - Recording
  - Indexing

- MySQL, PSQL proxy

- HTTP/HTTPs Web applications proxy:

All of the customer’s traffic is proxied via the IDAC

– SSH , RDP proxy:

  - Recording
  - Indexing

- MySQL, PSQL proxy

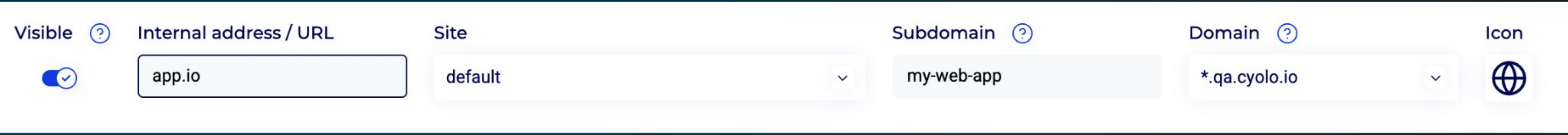
- HTTP/HTTPs Web applications proxy:
  - request/response transformations

HTTP/HTTPs Web applications proxy:

- application creation:

HTTP/HTTPs Web applications proxy:

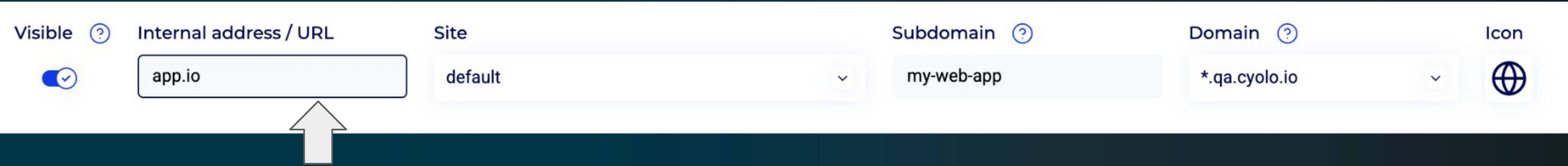
- application creation:



Visible	Internal address / URL	Site	Subdomain	Domain	Icon
✓	app.io	default	my-web-app	*.qa.cyolo.io	🌐

HTTP/HTTPs Web applications proxy:

- application creation:

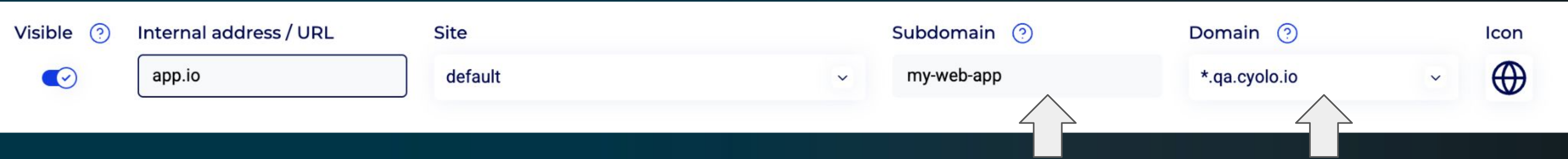


Visible	Internal address / URL	Site	Subdomain	Domain	Icon
✓	app.io	default	my-web-app	*.qa.cyolo.io	🌐

- internal address - the address of the application.

HTTP/HTTPs Web applications proxy:

- application creation:

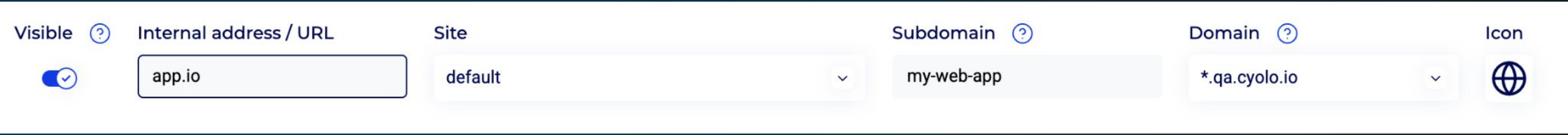


Visible	Internal address / URL	Site	Subdomain	Domain	Icon
✓	app.io	default	my-web-app	*.qa.cyolo.io	

  - <u>internal address</u> - the address of the application.
  - <u>subdomain+domain</u> - together make up the FQDN of the application.

HTTP/HTTPs Web applications proxy:

- application creation:



Visible	Internal address / URL	Site	Subdomain	Domain	Icon
✓	app.io	default	my-web-app	*.qa.cyolo.io	

- internal address - the address of the application.
- subdomain+domain - together make up the FQDN of the application.

external FQDN: my-web-app.qa.cyolo.io

HTTP/HTTPs Web applications proxy:

Example:

HTTP/HTTPs Web applications proxy:

Example:

Application 1:

- internal address: app1.io
- external address: app1.cyolo.io

HTTP/HTTPs Web applications proxy:

Example:

Application 1:

- internal address: app1.io
- external address: app1.cyolo.io

Application 2:

- internal address: app2.io
- external address: app2.cyolo.io

HTTP/HTTPs Web applications proxy:

Amit

HTTP/HTTPs Web applications proxy:

Amit

IDAC

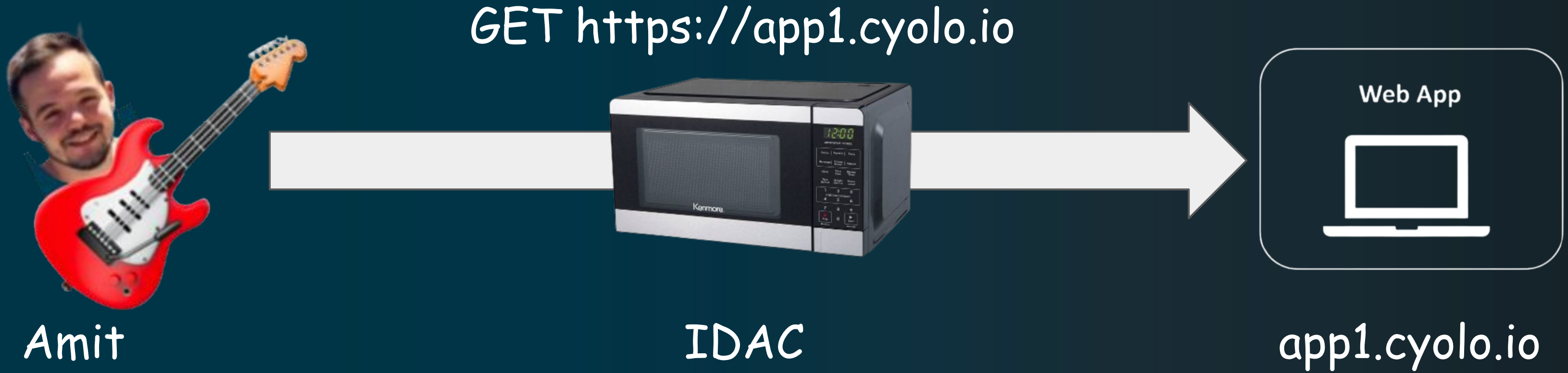
HTTP/HTTPs Web applications proxy:

Amit

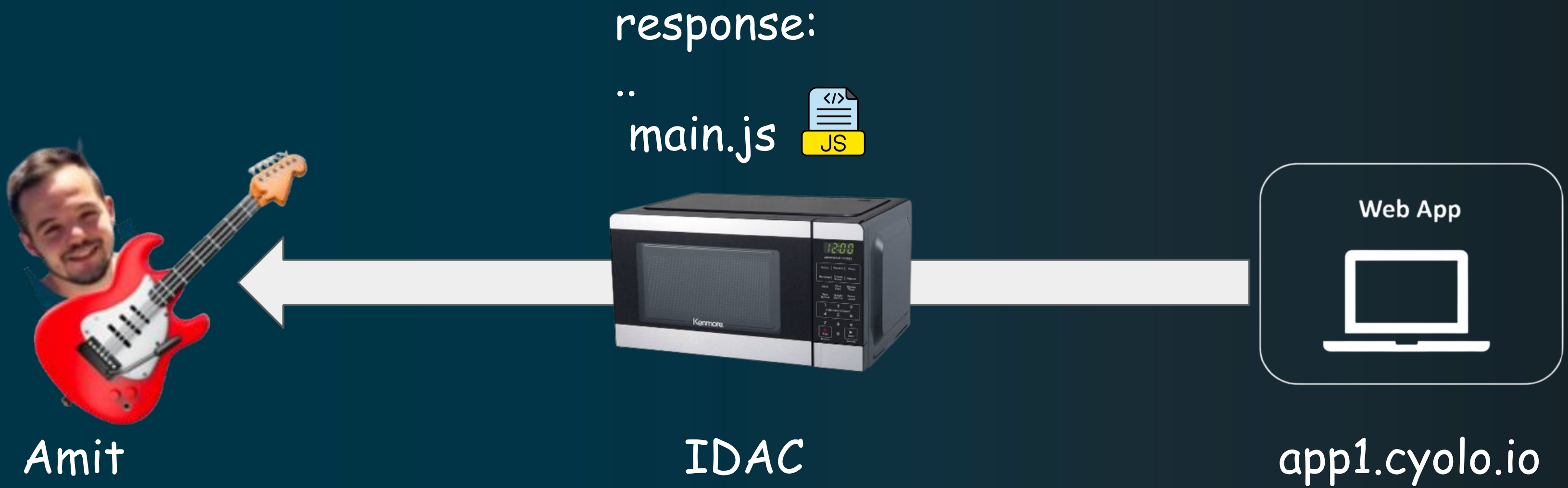
IDAC

app1.cyolo.io

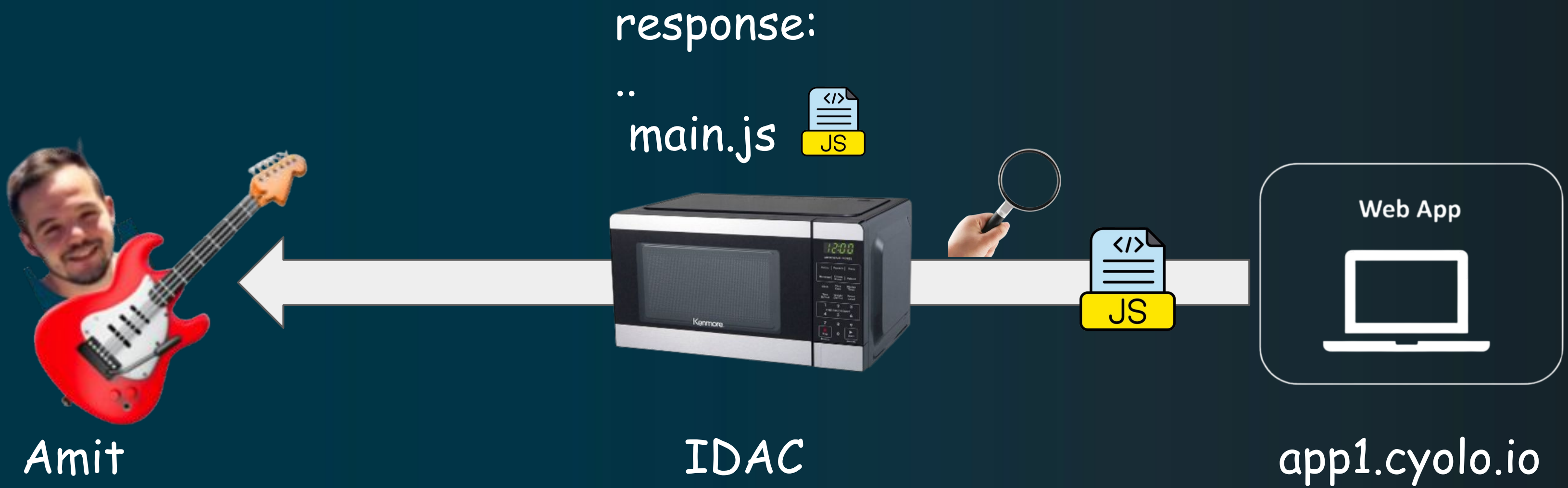
HTTP/HTTPs Web applications proxy:



HTTP/HTTPs Web applications proxy:



HTTP/HTTPs Web applications proxy:

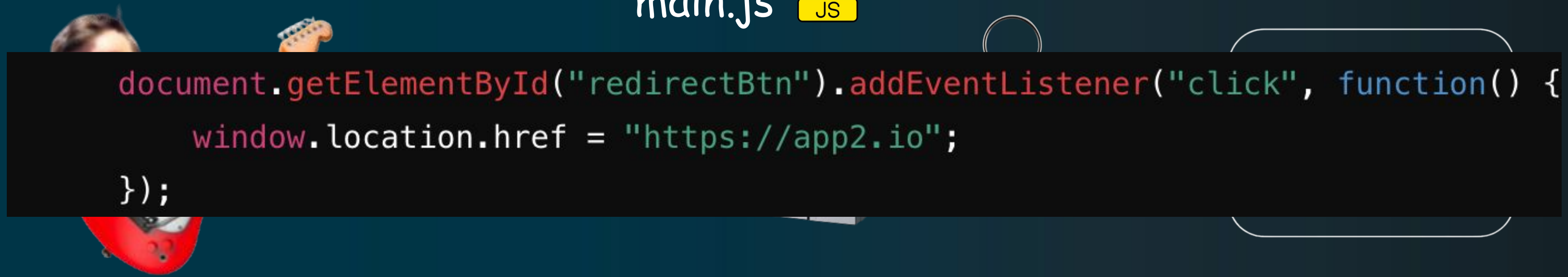


HTTP/HTTPs Web applications proxy:

response:

..

main.js



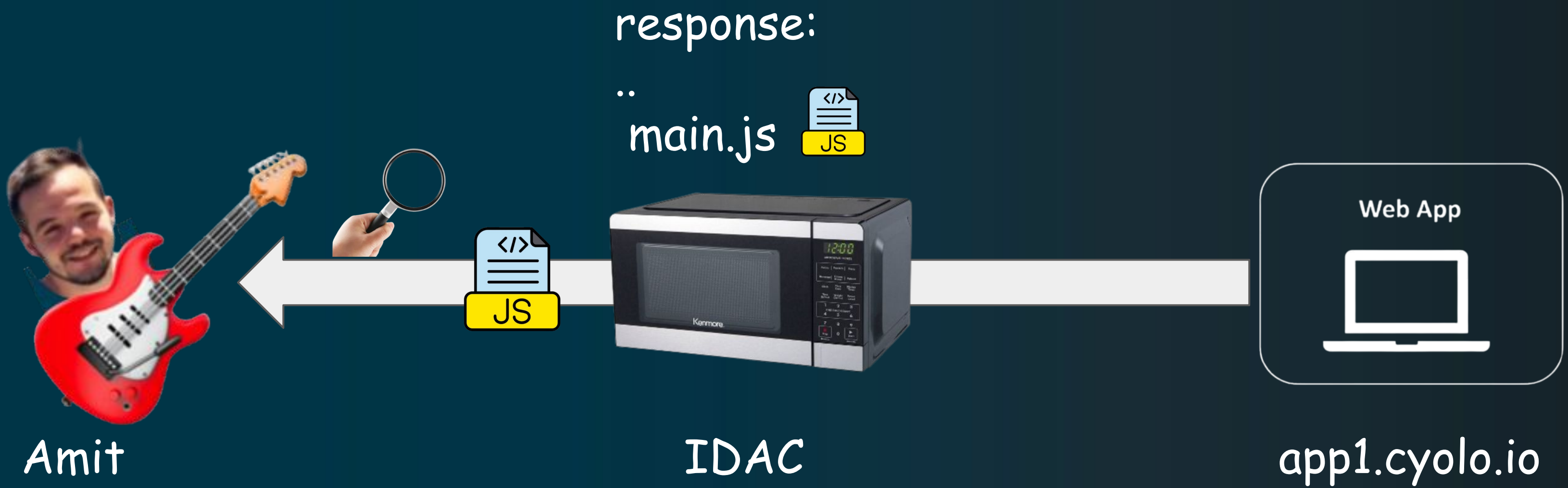
```
    document.getElementById("redirectBtn").addEventListener("click", function() {
        window.location.href = "https://app2.io";
    });
```

Amit

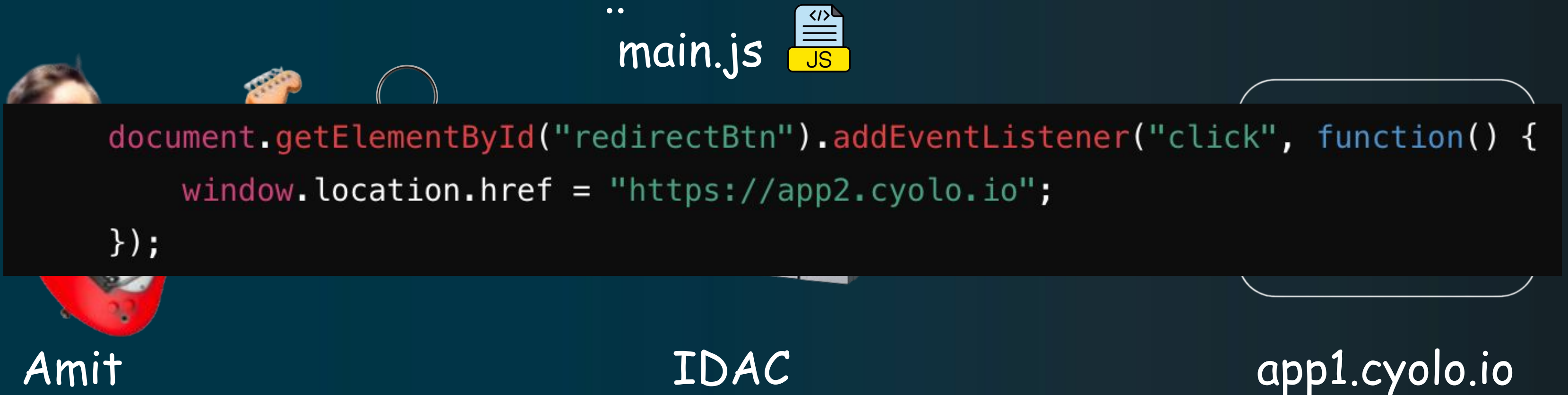
IDAC

app1.cyolo.io

HTTP/HTTPs Web applications proxy:



HTTP/HTTPs Web applications proxy:



```
    document.getElementById("redirectBtn").addEventListener("click", function() {
        window.location.href = "https://app2.cyolo.io";
    });
```

The objective:

- Rewrite response payload URLs with Cyolo’s domain.

The objective:

- Rewrite response payload URLs with Cyolo’s domain.

  NOTE: Hostnames are case insensitive!

The objective:

- Rewrite response payload URLs with Cyolo's domain.

  NOTE: Hostnames are case insensitive!

and for that we use:

The objective:

- Rewrite response payload URLs with Cyolo’s domain.

and for that we use:

TRANSFOMERS

The objective:

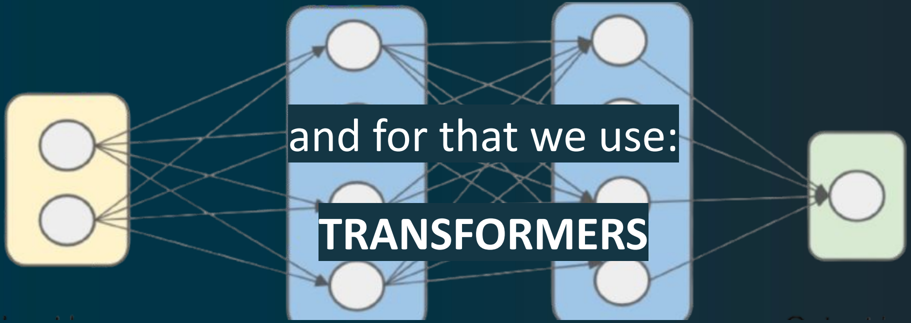
- Rewrite response payload URLs with Cyolo’s domain.

and for that we use:

TRANSFOMERS

The objective:

- Rewrite response payload URLs with Cyolo's domain.

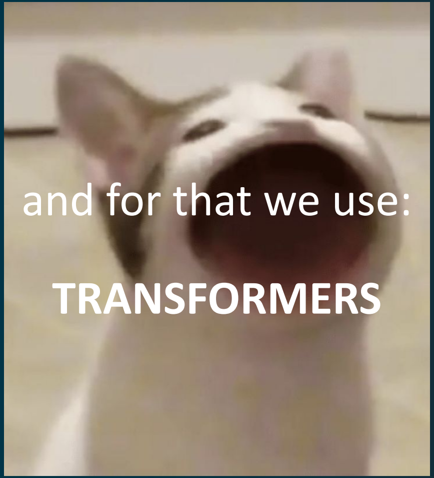


and for that we use:

TRANSFORMERS

The objective:

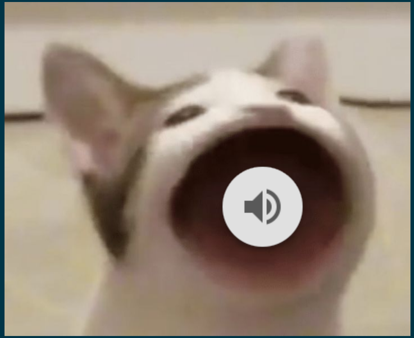
- Rewrite response payload URLs with Cyolo’s domain.



and for that we use:

TRANSFORMERS

Prerequisite:



- New professional jargon from now on:

MACHTUL

Prerequisite:

- New professional jargon from now on:

MACHTUL

My **A**pplication **C**rashes **H**ard, **T**hrows **U**nexpected **L**ogs

Prerequisite:

- New professional jargon from now on:

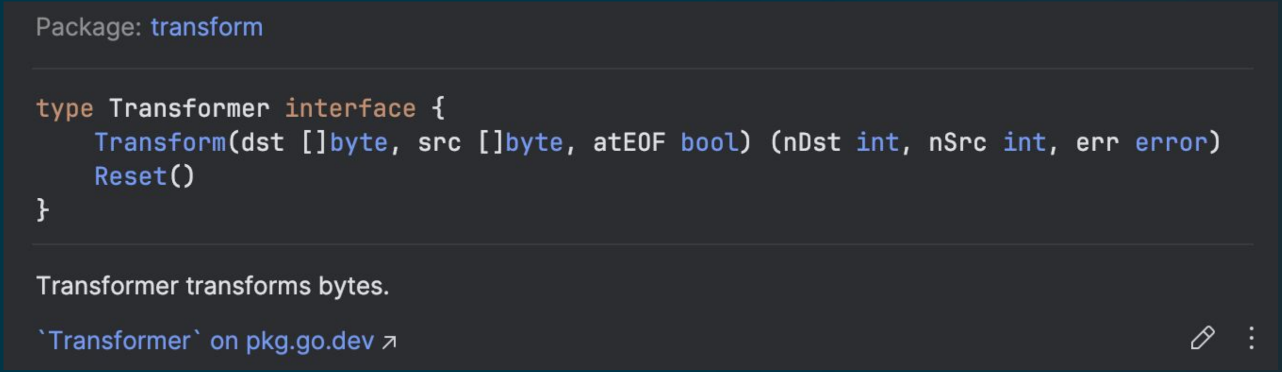
MACHTUL

My **A**pplication **C**rashes **H**ard, **T**hrows **U**nexpected **L**ogs

When you see this cat you know that something went very wrong!

Transformers:

golang.org/x/text/transform



Package: transform

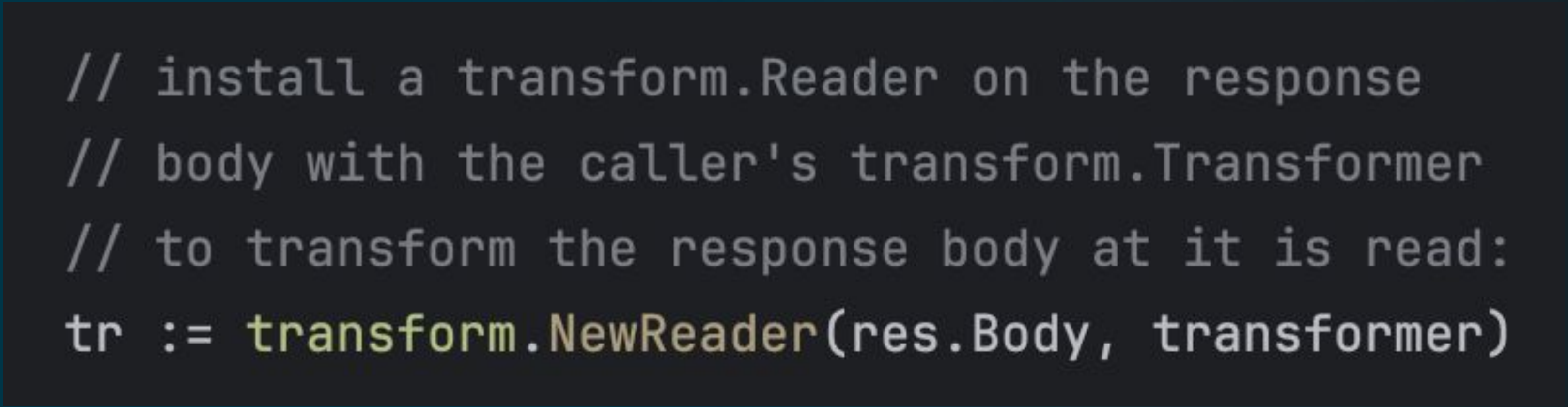
```
type Transformer interface {
    Transform(dst []byte, src []byte, atEOF bool) (nDst int, nSrc int, err error)
    Reset()
}
```

Transformer transforms bytes.

`Transformer` on pkg.go.dev ↗

- Transform writes to dst the transformed bytes read from src.

Transformers:



```go
// install a transform.Reader on the response
// body with the caller's transform.Transformer
// to transform the response body at it is read:
tr := transform.NewReader(res.Body, transformer)
```

1st attempt:

Naive chunk-based transformation

Naive Chunk based transformation:

How?

Naive Chunk based transformation:

How?

- Read main.js file one chunk at a time.

Naive Chunk based transformation:

How?

- Read main.js file one chunk at a time.

- Lower case everything.

Naive Chunk based transformation:

How?

- Read main.js file one chunk at a time.

- Lower case everything.

- IDAC has the mapping between application internal address -> external address.

Naive Chunk based transformation:

How?

- Read main.js file one chunk at a time.
- Lower case everything.
- IDAC has the mapping between application internal address -> external address.
- Replace app2.io with app2.cyolo.io .

Naive Chunk based transformation:

Before:

```
… window.location.href = “https://a
```

CHUNK N

```
pp2.io”; …
```

CHUNK N+1

Naive Chunk based transformation:

Before:

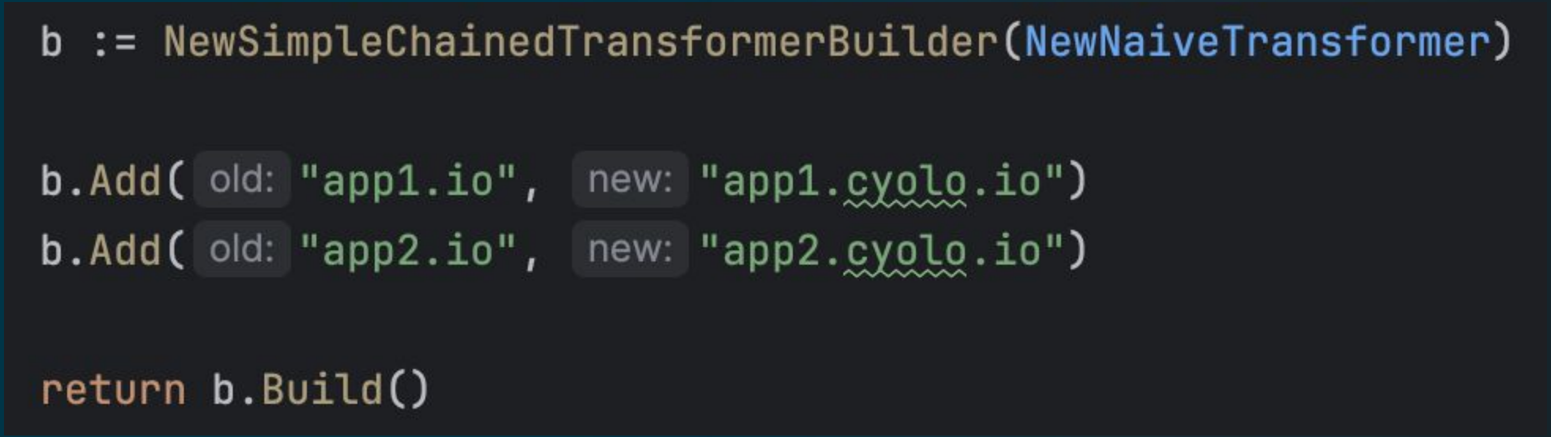
… window.location.href = “https://a	pp2.io”; …
CHUNK N	CHUNK N+1

After:

… window.location.href = “https://a	pp2.cyolo.io”; …
CHUNK N	CHUNK N+1

Naive Chunk based transformation:

Building:



```
b := NewSimpleChainedTransformerBuilder(NewNaiveTransformer)

b.Add( old: "app1.io",  new: "app1.cyolo.io")
b.Add( old: "app2.io",  new: "app2.cyolo.io")

return b.Build()
```

Naive Chunk based transformation:

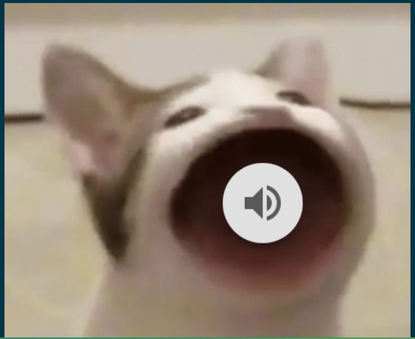
- This solution ran in production for quite some time..

Naive Chunk based transformation:

- This solution ran in production for quite some time..

Machtul panic:

```
panic: runtime error: index out of range
```



Naive Chunk based transformation:

Short QUIZ:

str := “ApP2.Io”

len(str) = **?**

Naive Chunk based transformation:

Short QUIZ:

str := “ApP2.Io”

len(str) = **7**

Naive Chunk based transformation:

Short QUIZ:

```
str := “ApP2.Io”

len(strings.ToLower(str)) = ?
```

Naive Chunk based transformation:

Short QUIZ:

str := “ApP2.Io”

len(strings.ToLower(str)) = **7**

Naive Chunk based transformation:

Short QUIZ:

str := “A̸pP2.Io”

len(str) = **?**

Naive Chunk based transformation:

Short QUIZ:

str := “ȺpP2.Io”

len(str) = **8**

Naive Chunk based transformation:

Short QUIZ:

str := “ȺpP2.Io”

len(strings.ToLower(str)) = ?

Naive Chunk based transformation:

Short QUIZ:

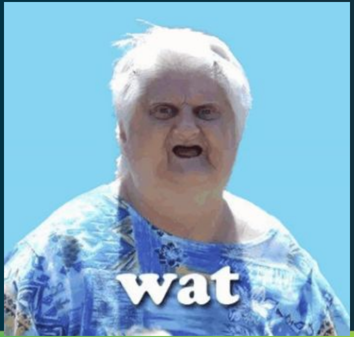
```
str := “ȺpP2.Io”

len(strings.ToLower(str)) = 9
```

Naive Chunk based transformation:

Short QUIZ:

str := “ȺpP2.Io”

len(strings.ToLower(str)) = **9**



Naive Chunk based transformation:

- strings are stream of bytes of UTF-8 encoded characters.

Naive Chunk based transformation:

- strings are stream of bytes of UTF-8 encoded characters.

- UTF-8 is a variable-length encoding, from 1 to 4 bytes.

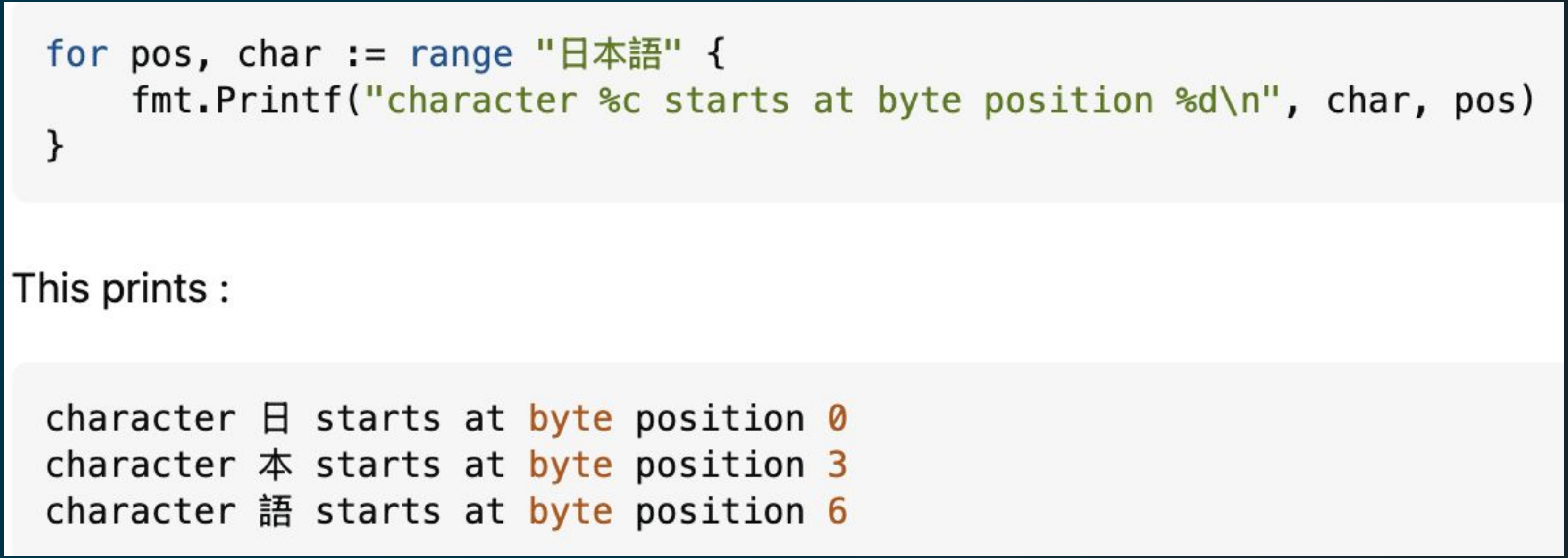
Naive Chunk based transformation:

- strings are stream of bytes of UTF-8 encoded characters.

- UTF-8 is a variable-length encoding, from 1 to 4 bytes.

- rune is an alias to int32 (4 bytes) used to store a UTF-8 encoding.

Naive Chunk based transformation:



```
for pos, char := range "日本語" {
    fmt.Printf("character %c starts at byte position %d\n", char, pos)
}
```

This prints :

```
character 日 starts at byte position 0
character 本 starts at byte position 3
character 語 starts at byte position 6
```

Naive Chunk based transformation:

- This broke our assumption that it is OK to lowercase everything.

Naive Chunk based transformation:

- This broke our assumption that it is OK to lowercase everything.

- Leading to out of bounds access of the chunk buffer while scanning.

Naive Chunk based transformation:

How can we preserve the case insensitivity of our algorithm?

Naive Chunk based transformation:

How can we preserve the case insensitivity of our algorithm?

- Naively keeping all lower/upper combinations.

aPp2.Io

App2.iO —> app2.cyolo.io

.

.

Naive Chunk based transformation:

- Extremely inefficient in terms of space and time.

Naive Chunk based transformation:

- Extremely inefficient in terms of space and time.

- Which led us to our 2nd attempt.

2nd attempt:

Regex

Regex transformer:

```
return replace.RegexpString(regexp.MustCompile("(?i)"+regexp.QuoteMeta(old)), new)
```

- one liner that creates a struct that implements transform.Transformer.

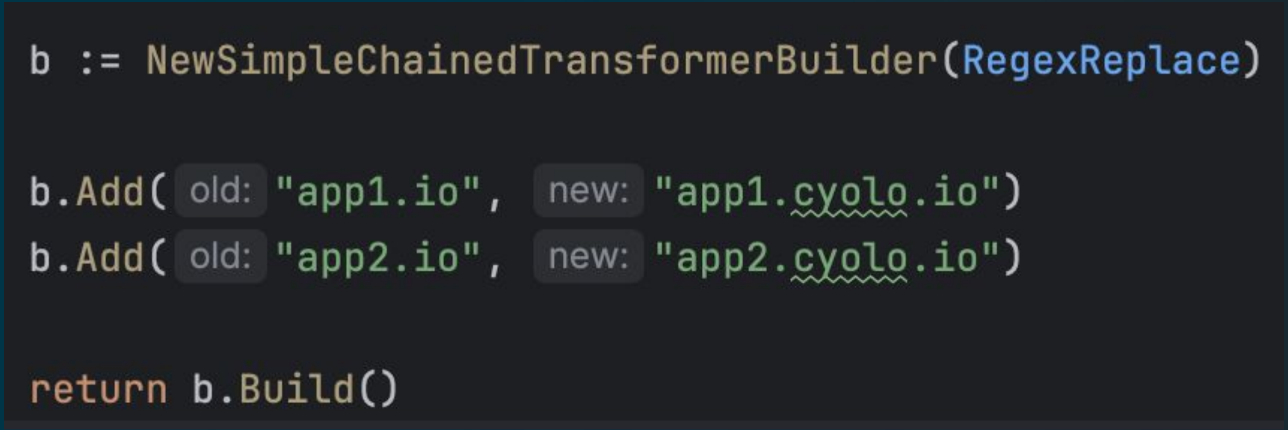
- first argument is the to-be internal address that should be replaced.

- (?i) makes the regex case-insensitive which is exactly what we need.

- handles internally the unicode problem we had earlier.

Regex transformer:

Building:



```
b := NewSimpleChainedTransformerBuilder(RegexReplace)

b.Add( old: "app1.io",  new: "app1.cyolo.io")
b.Add( old: "app2.io",  new: "app2.cyolo.io")

return b.Build()
```

Regex transformer:

- after some local testing, it seemed to worked fine.

- a patch was shipped, everyone’s happy.

Regex transformer:

- after some local testing, it seemed to worked fine.

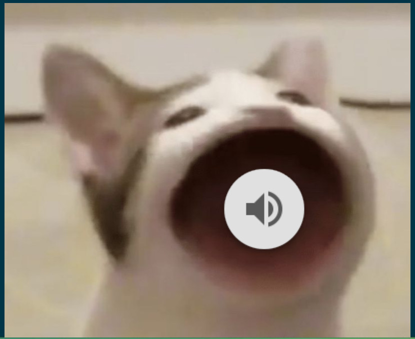
- a patch was shipped, everyone’s happy.

Until….

Regex transformer:

- after some local testing, it seemed to worked fine.

- a patch was shipped, everyone’s happy.



Machtulatency:

Meow, 20mb JS file takes 30 seconds to process!!

Regex transformer:

```
BenchmarkResponseRewriteTransformerForMappings/regex_many-11
1    30001756542 ns/op          30002 ms/op    128591352 B/op      515 allocs/op
```

Regex transformer:

- There is no choice but to come up with a top-notch solution.

Regex transformer:

- There is no choice but to come up with a top-notch solution.

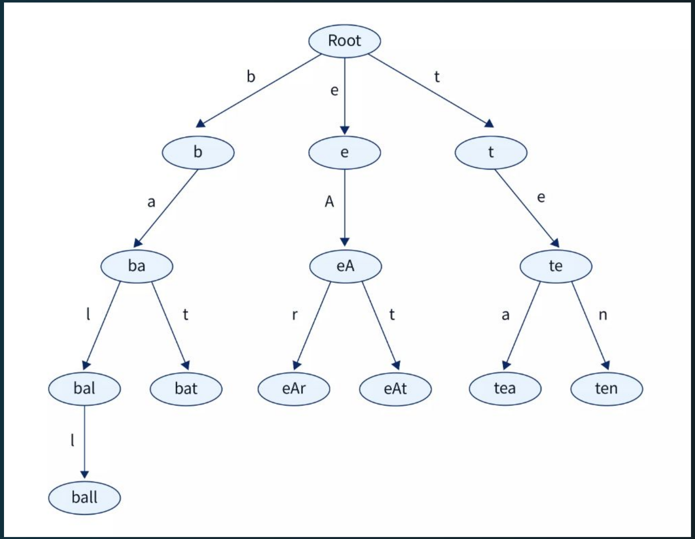
- Which led us to the 3rd and final attempt.

3rd attempt:

The Trie transformer

The Trie transformer:

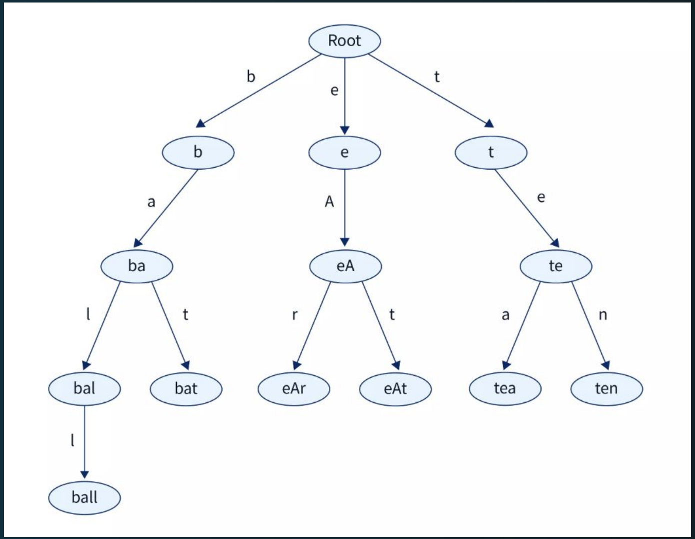
What is the Trie (prefix tree) data structure?



The Trie transformer:

What is the Trie (prefix tree) data structure?

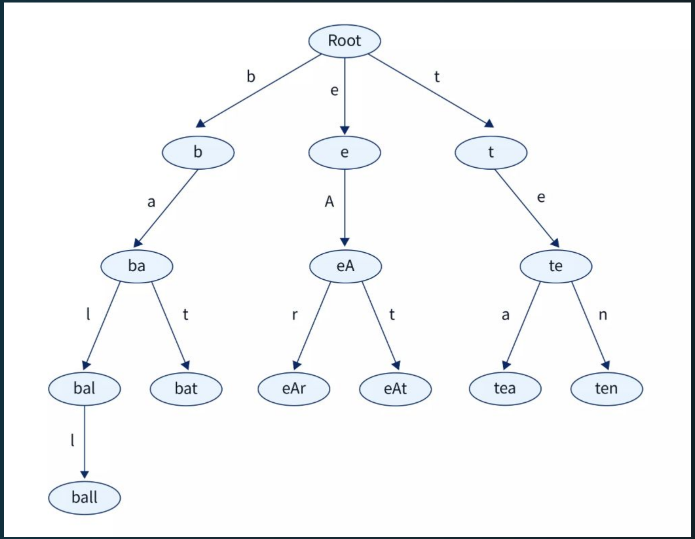
- N-ary tree that is efficient for string matching.



The Trie transformer:

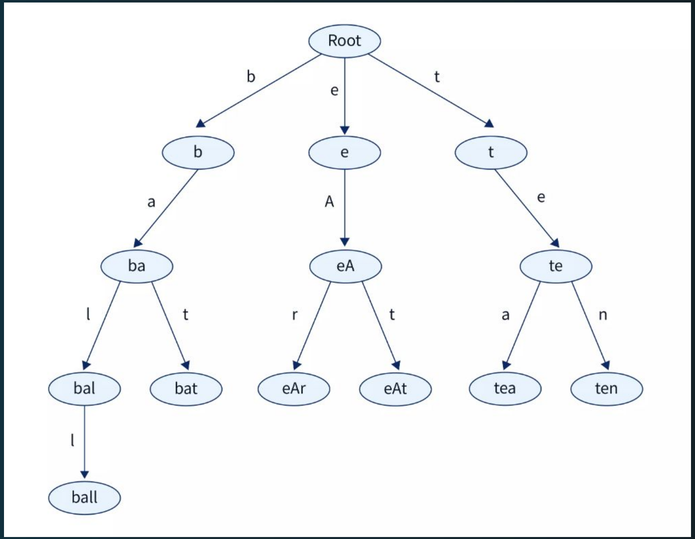
What is the Trie (prefix tree) data structure?

- N-ary tree that is efficient for string matching.
- Each traversal from root to leaf is a string we store in order to match.



The Trie transformer:

What is the Trie (prefix tree) data structure?

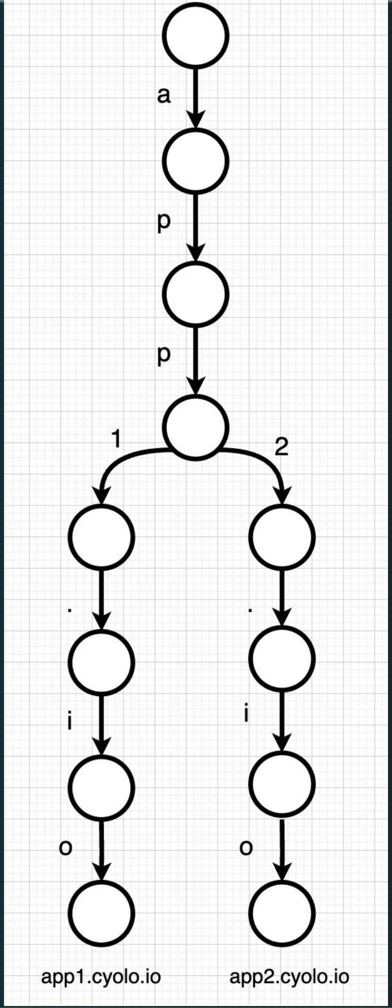


- N-ary tree that is efficient for string matching.
- Each traversal from root to leaf is a string we store in order to match.
- Strings that share prefixes will share the same sub-route in the tree, which makes the storage efficient by avoiding duplicates.

The Trie transformer:

In our use case:

- inserted:
  - app1.io
  - app2.io

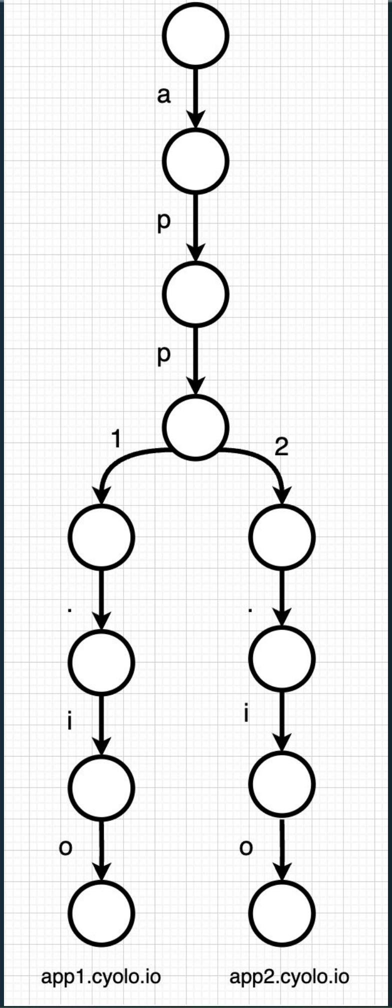


The Trie transformer:

In our use case:

- inserted:
    - app1.io
    - app2.io

- t.Match(“app1.io”)

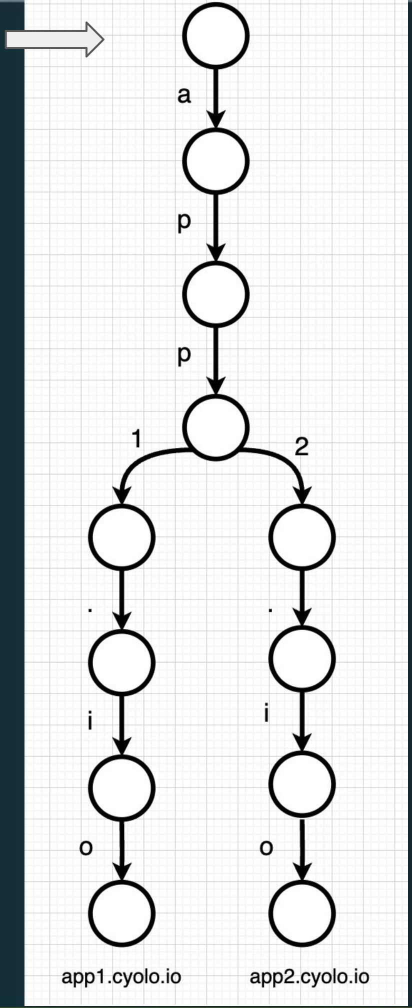


The Trie transformer:

In our use case:

- inserted:
  - app1.io
  - app2.io

- t.Match(“app1.io”)

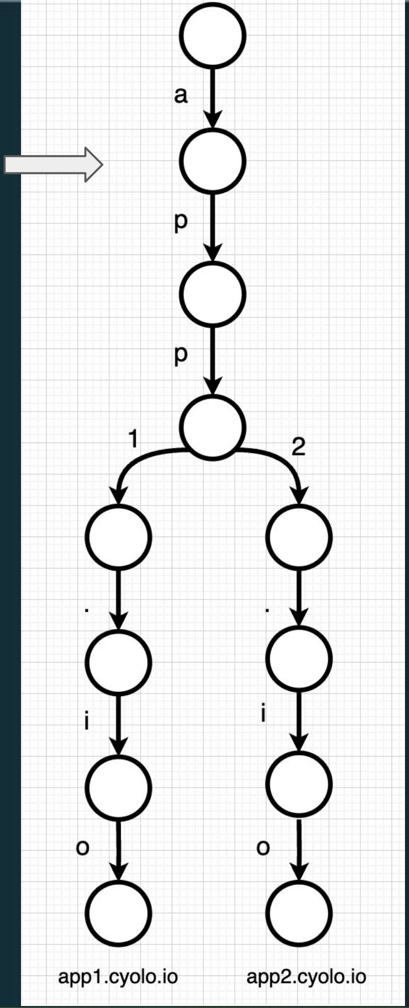


The Trie transformer:

In our use case:

- inserted:
    - app1.io
    - app2.io

- t.Match(“app1.io”)

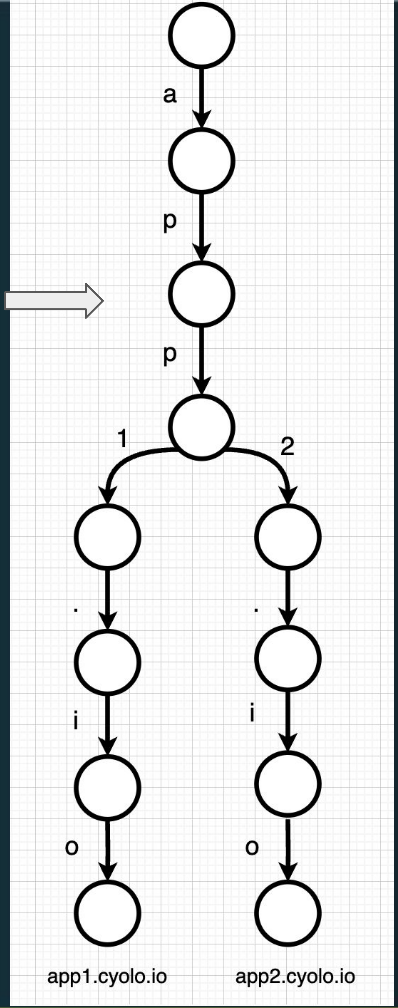


The Trie transformer:

In our use case:

- inserted:
  - app1.io
  - app2.io

- t.Match(“app1.io”)

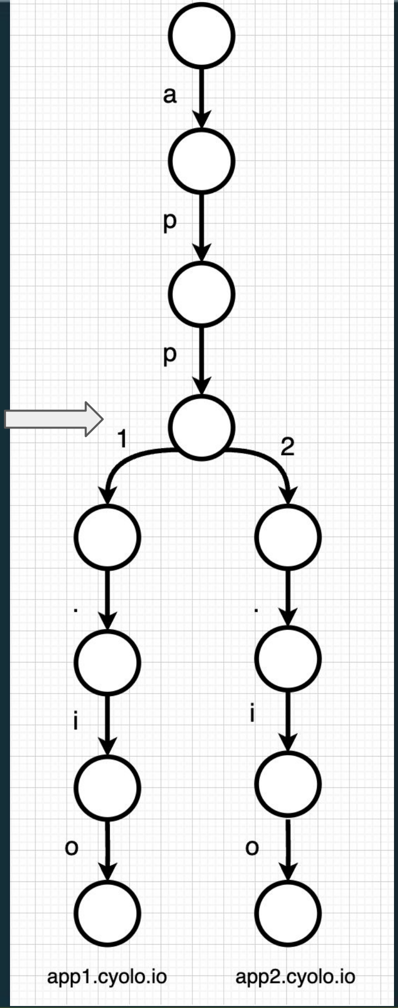


The Trie transformer:

In our use case:

- inserted:
    - app1.io
    - app2.io

- t.Match(“app1.io”)

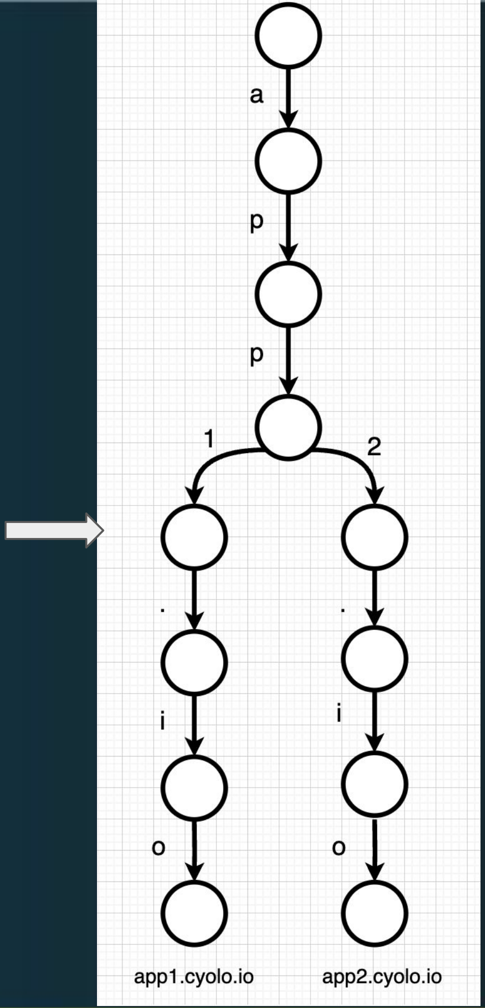


The Trie transformer:

In our use case:

- inserted:
  - app1.io
  - app2.io

- t.Match(“app1.io”)

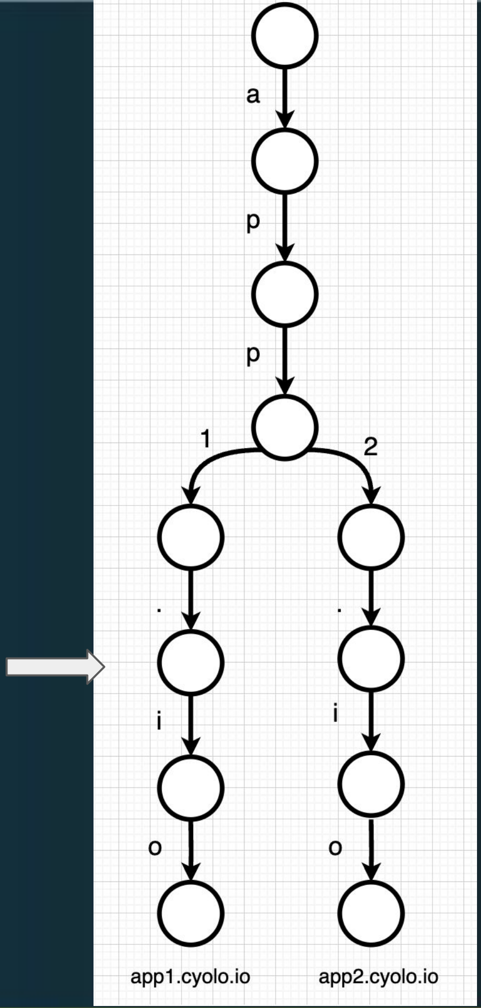


The Trie transformer:

In our use case:

- inserted:
    - app1.io
    - app2.io

- t.Match(“app1.io”)

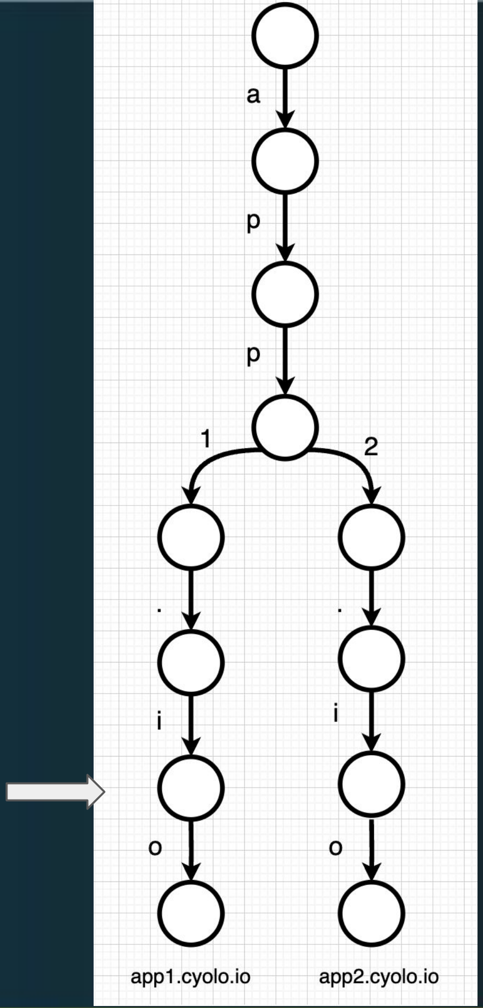


The Trie transformer:

In our use case:

- inserted:
  - app1.io
  - app2.io

- t.Match(“app1.io”)

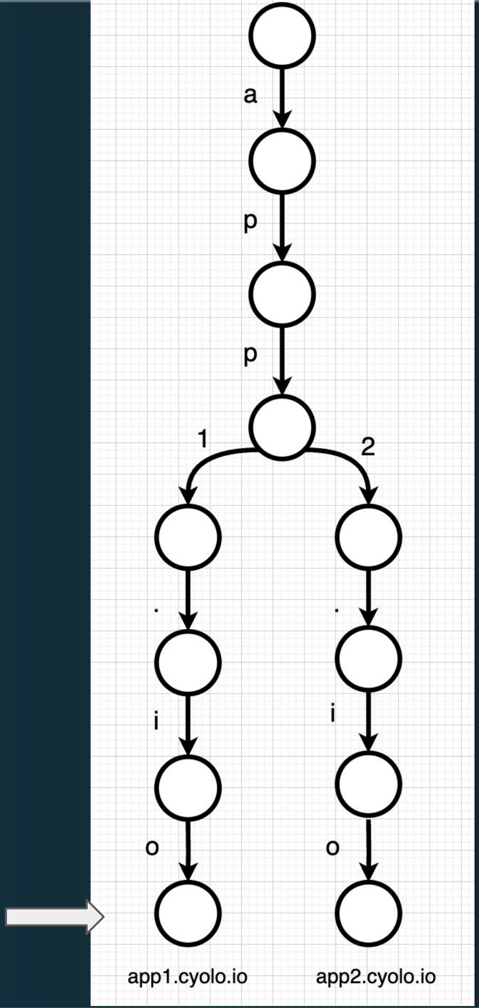


The Trie transformer:

In our use case:

- inserted:
    - app1.io
    - app2.io

- t.Match(“app1.io”)



The Trie transformer:

Preserving case-insensitivity remains a challenge!

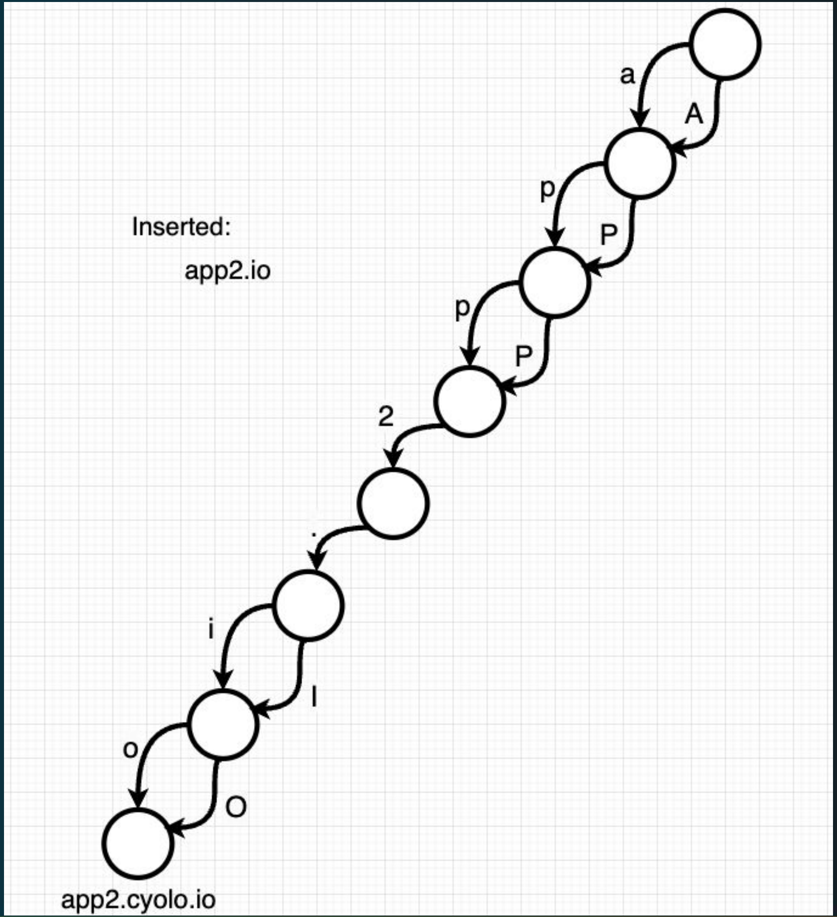
The Trie transformer:

Can we do better?

The Trie transformer:

Case insensitive trie (actually a DAG):

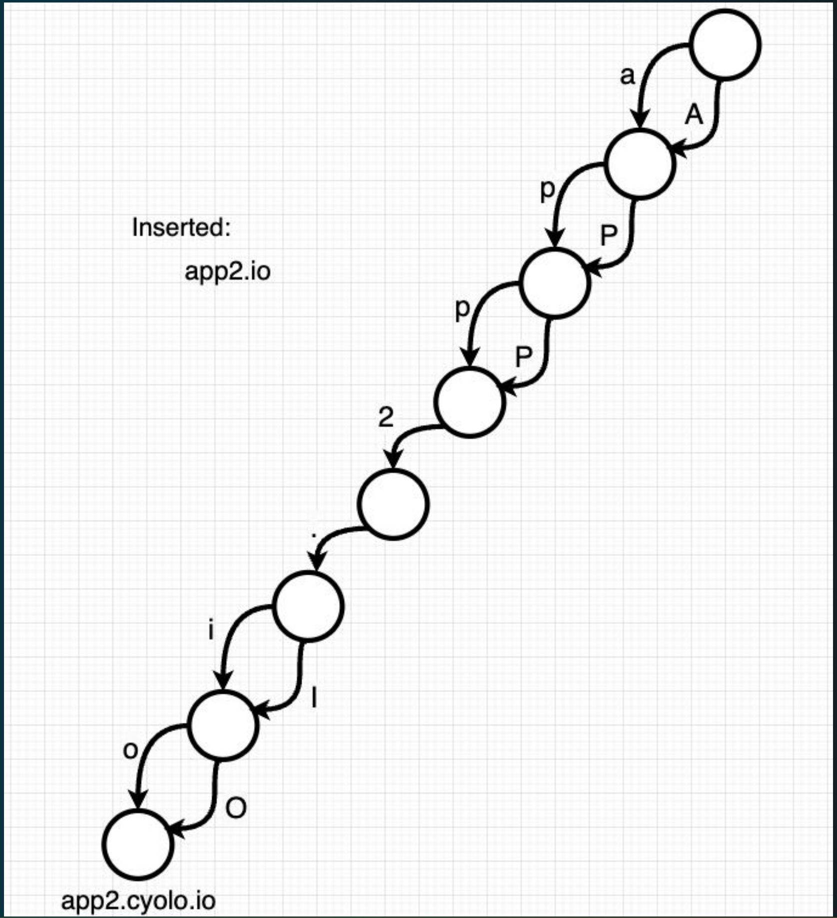
- upper/lower combinations will form the same route to leaf.



The Trie transformer:

Case insensitive trie (actually a DAG):

- upper/lower combinations will form the same route to leaf.
- t.Match(“ApP2.Io”)



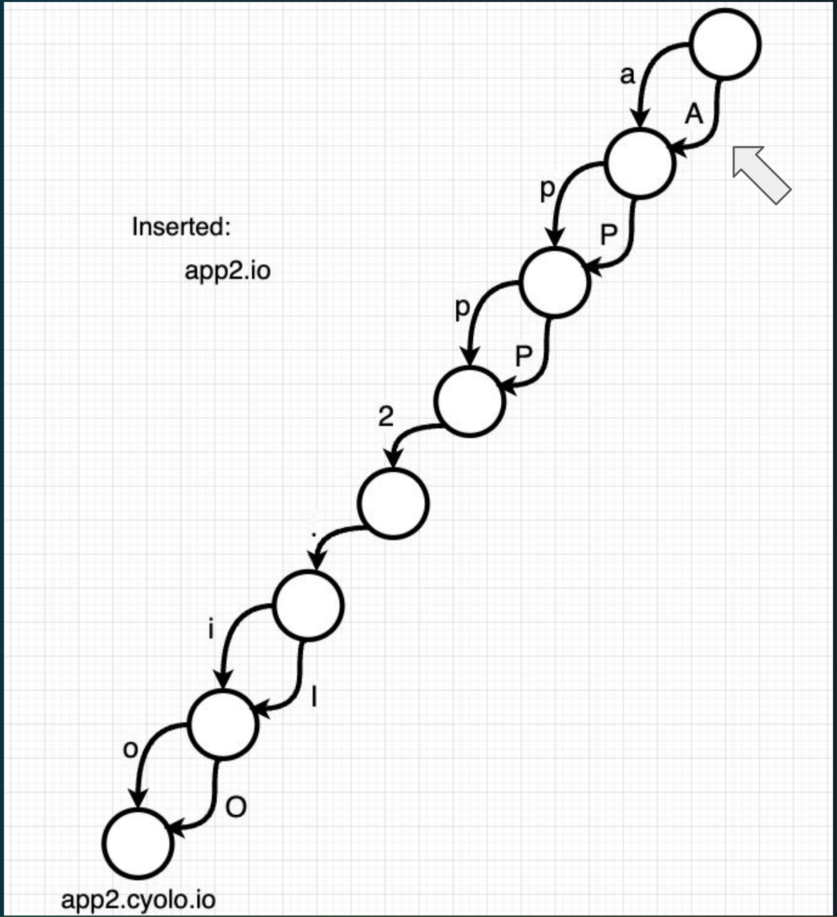
The Trie transformer:

Case insensitive trie (actually a DAG):

- upper/lower combinations will form

  the same route to leaf.

- t.Match(“ApP2.Io”)



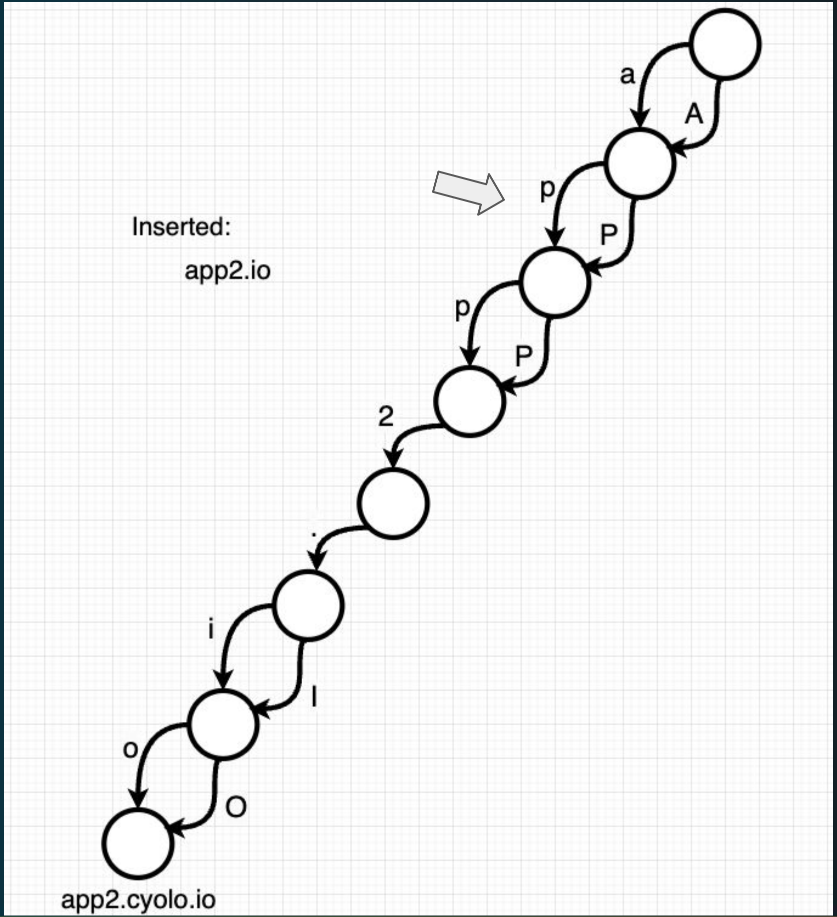
The Trie transformer:

Case insensitive trie (actually a DAG):

- upper/lower combinations will form

  the same route to leaf.

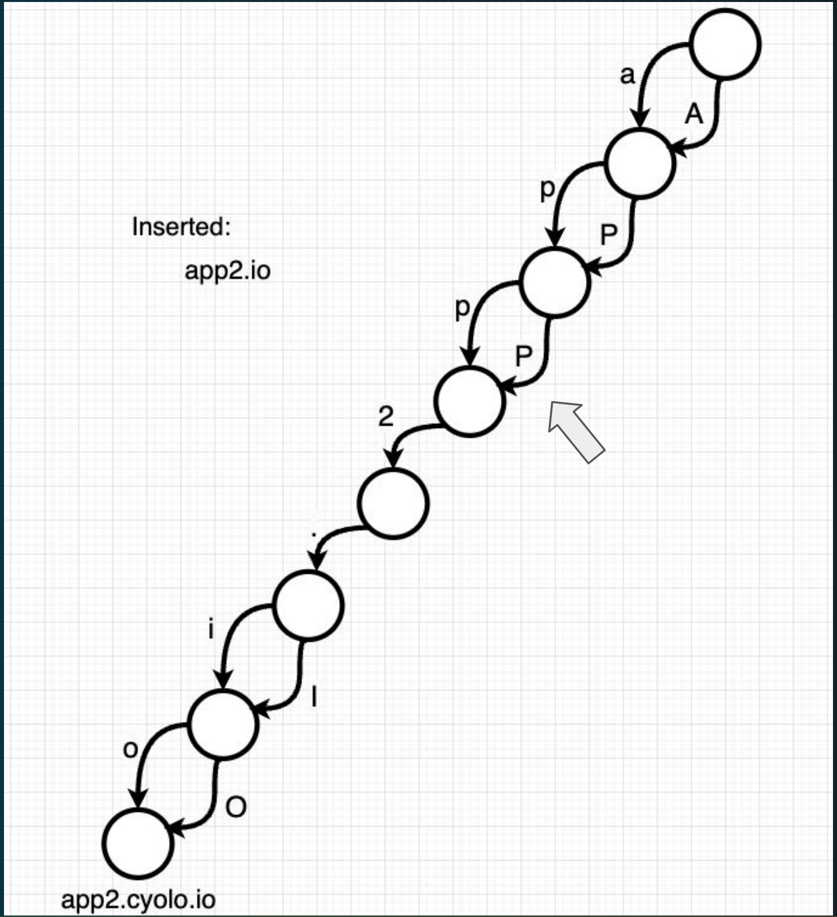
- t.Match(“ApP2.Io”)



The Trie transformer:

Case insensitive trie (actually a DAG):

- upper/lower combinations will form the same route to leaf.
- t.Match(“ApP2.Io”)

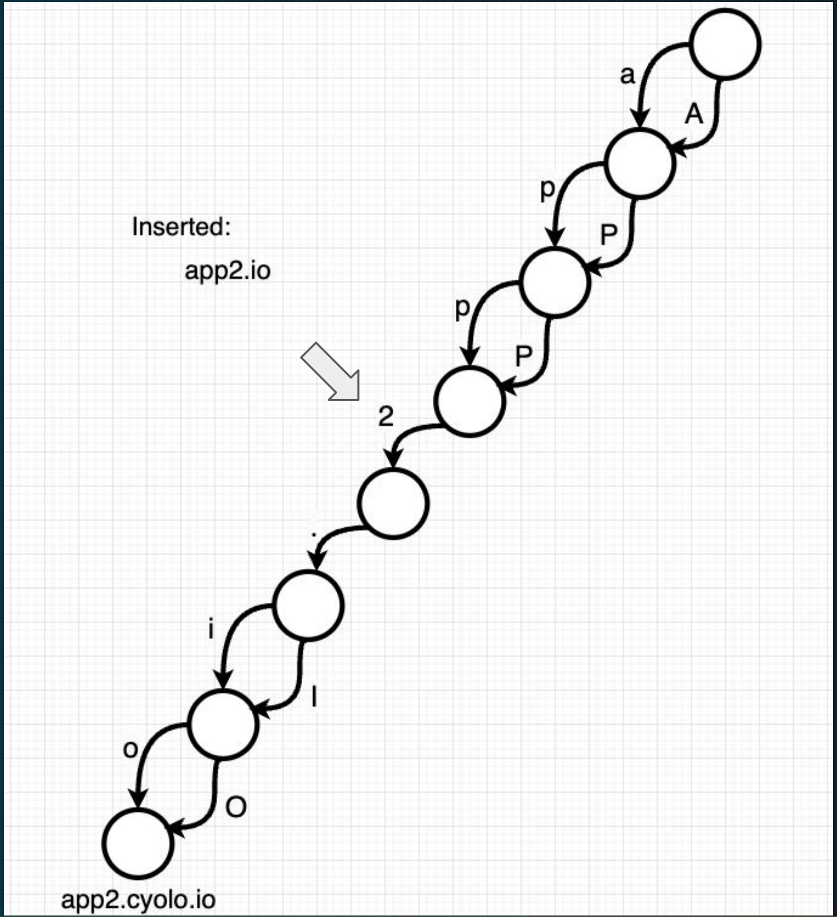


The Trie transformer:

Case insensitive trie (actually a DAG):

- upper/lower combinations will form the same route to leaf.

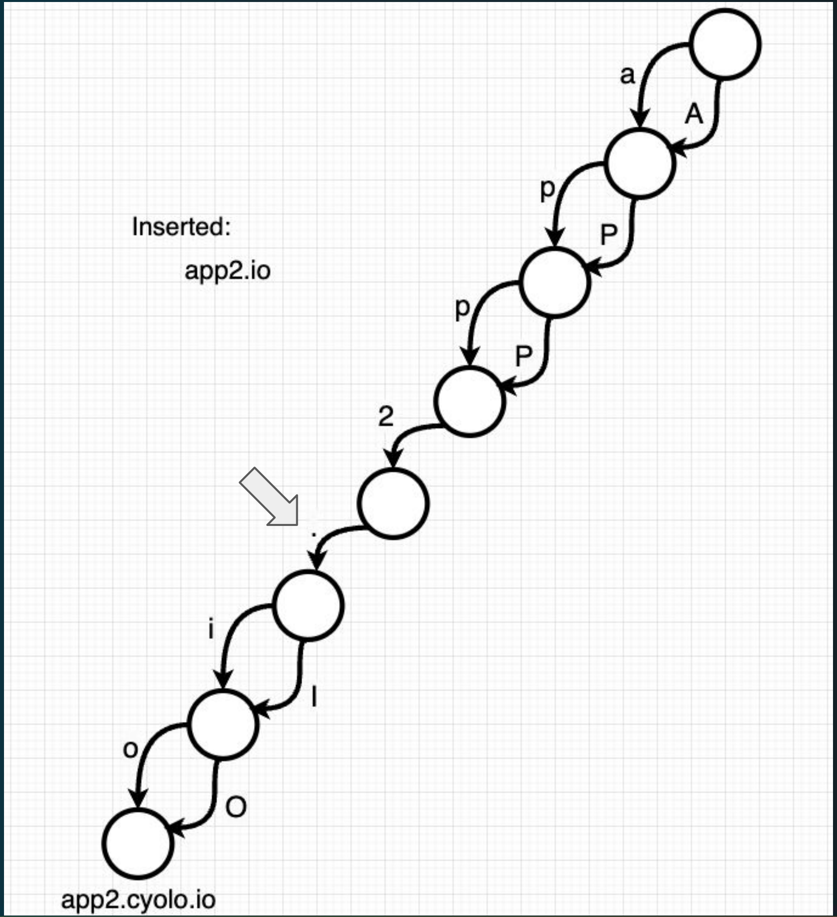
- t.Match("ApP2.Io")



The Trie transformer:

Case insensitive trie (actually a DAG):

- upper/lower combinations will form the same route to leaf.
- t.Match(“ApP2.Io”)

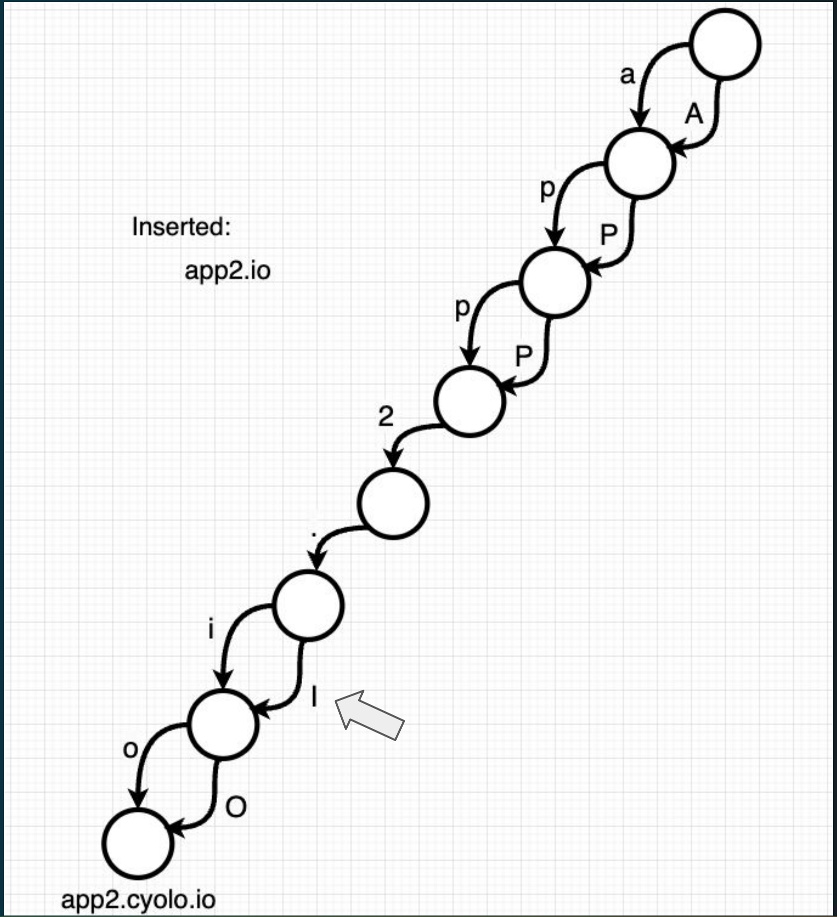


The Trie transformer:

Case insensitive trie (actually a DAG):

- upper/lower combinations will form the same route to leaf.

- t.Match(“ApP2.Io”)



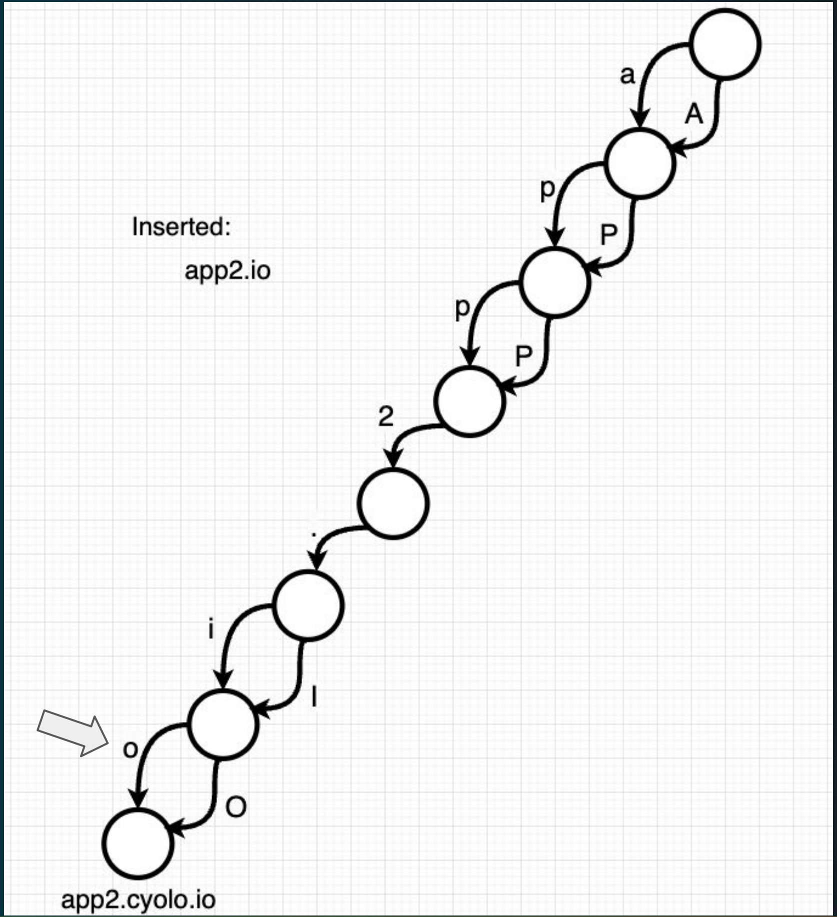
The Trie transformer:

Case insensitive trie (actually a DAG):

- upper/lower combinations will form

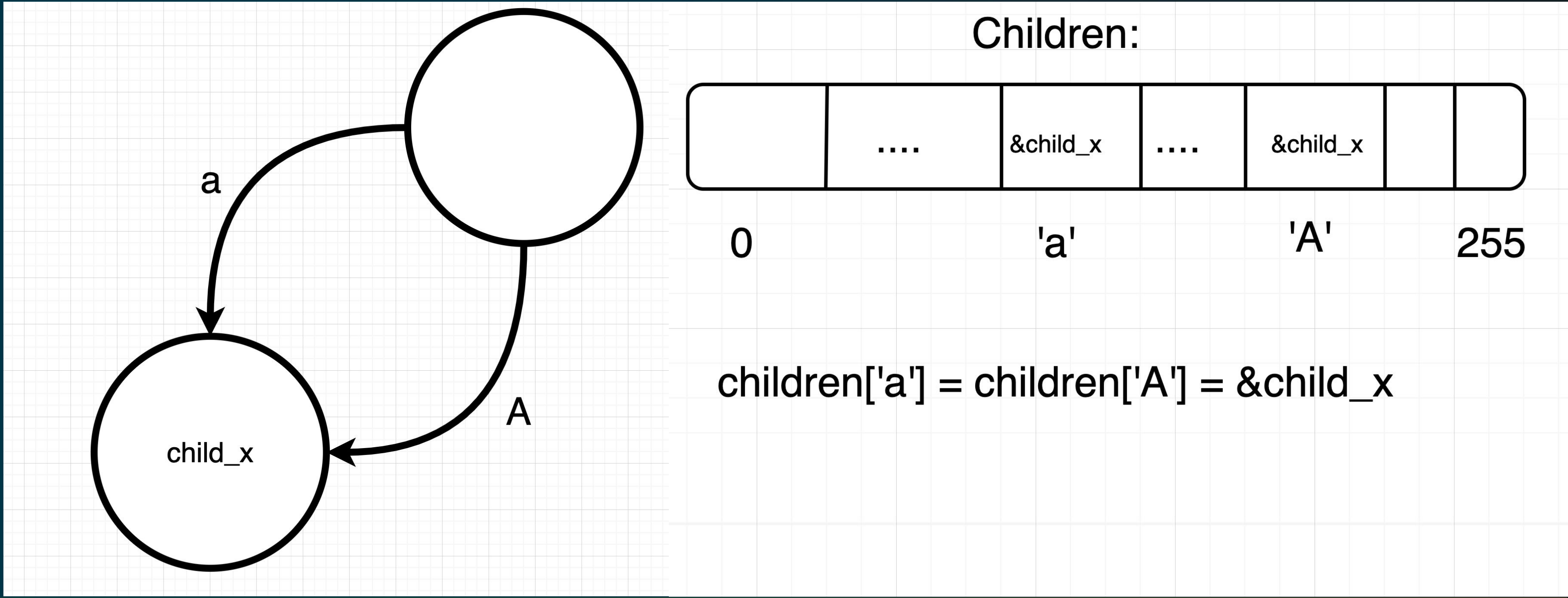
  the same route to leaf.

- t.Match("ApP2.Io")



The Trie transformer:

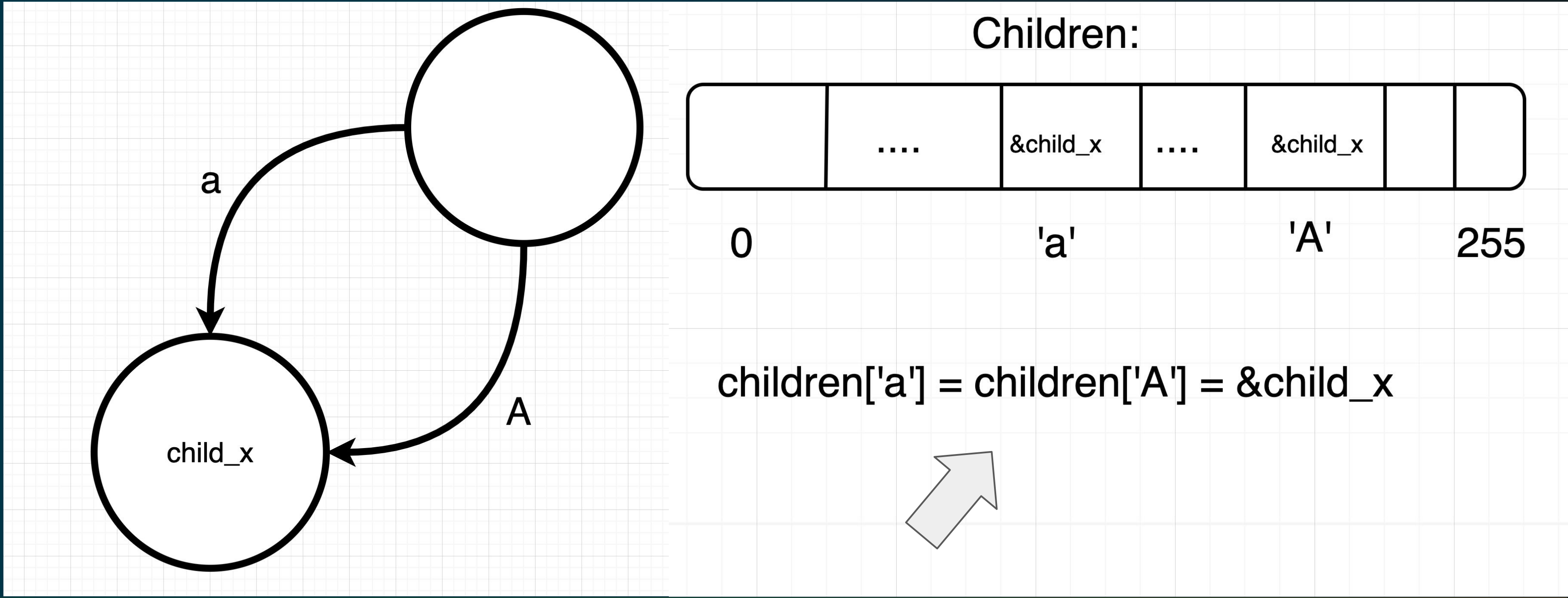
Case insensitive trie (actually a DAG):



children['a'] = children['A'] = &child_x

The Trie transformer:

Case insensitive trie (actually a DAG):

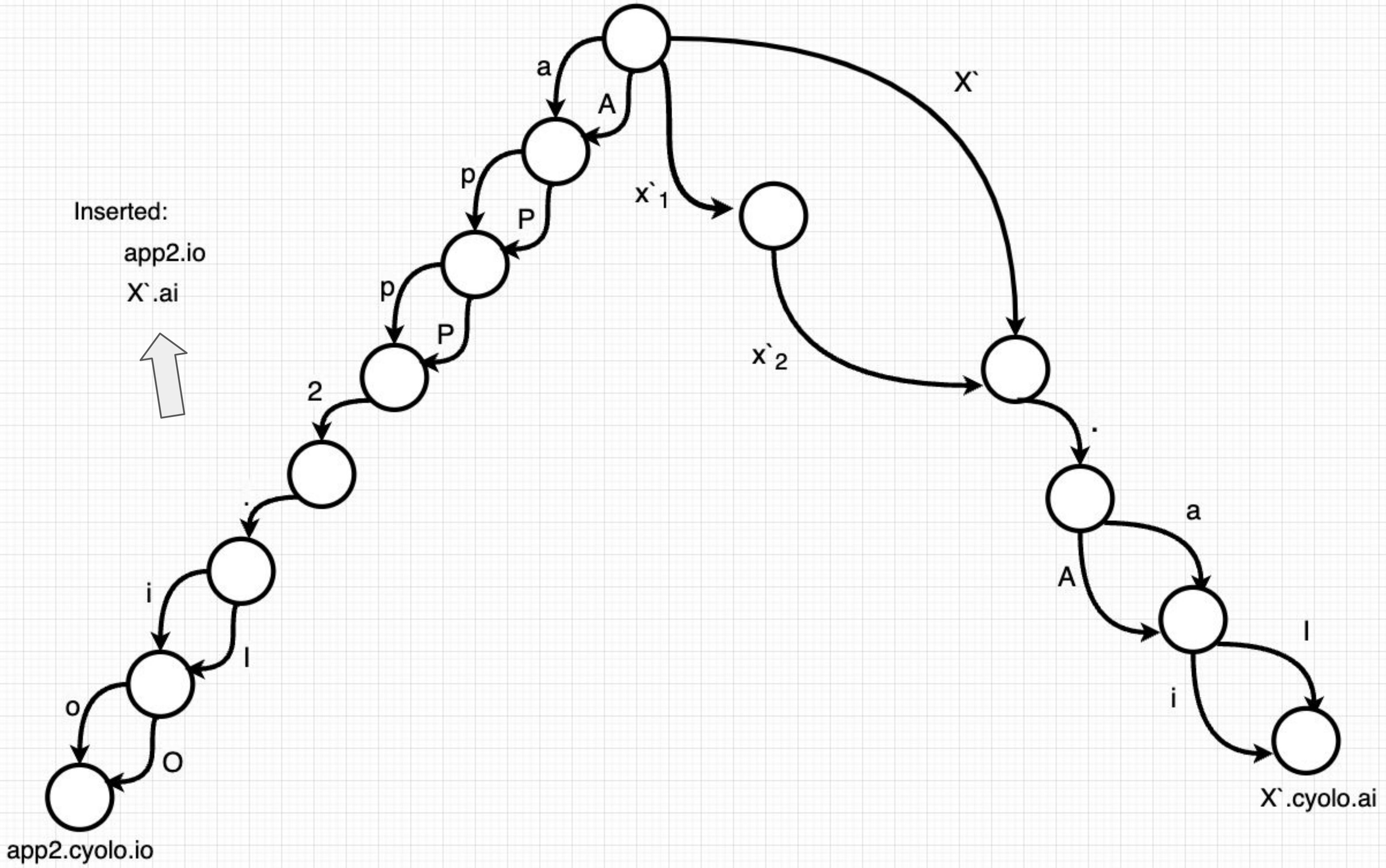


children['a'] = children['A'] = &child_x

Inserted:

app2.io

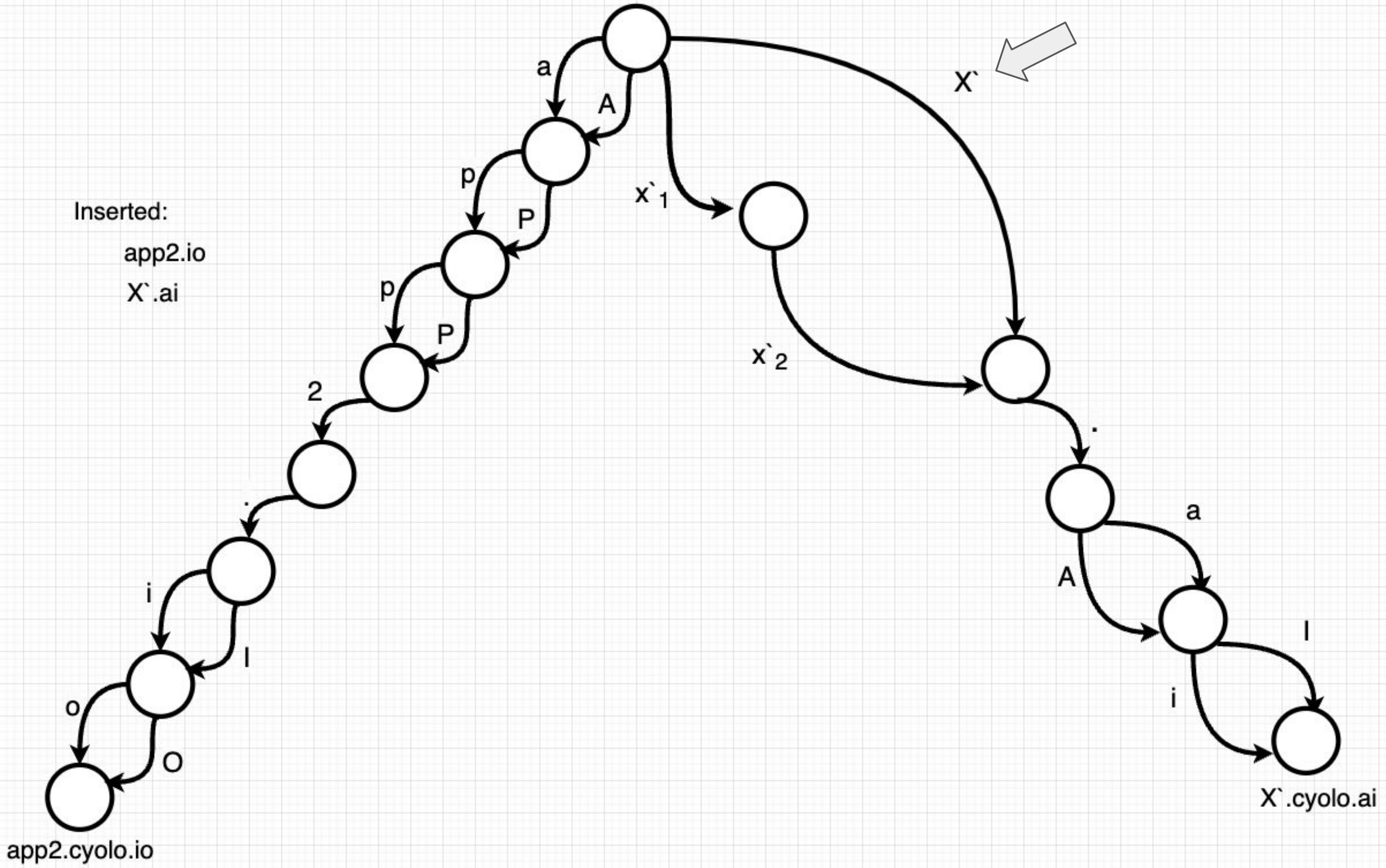
X`.ai



Inserted:

app2.io

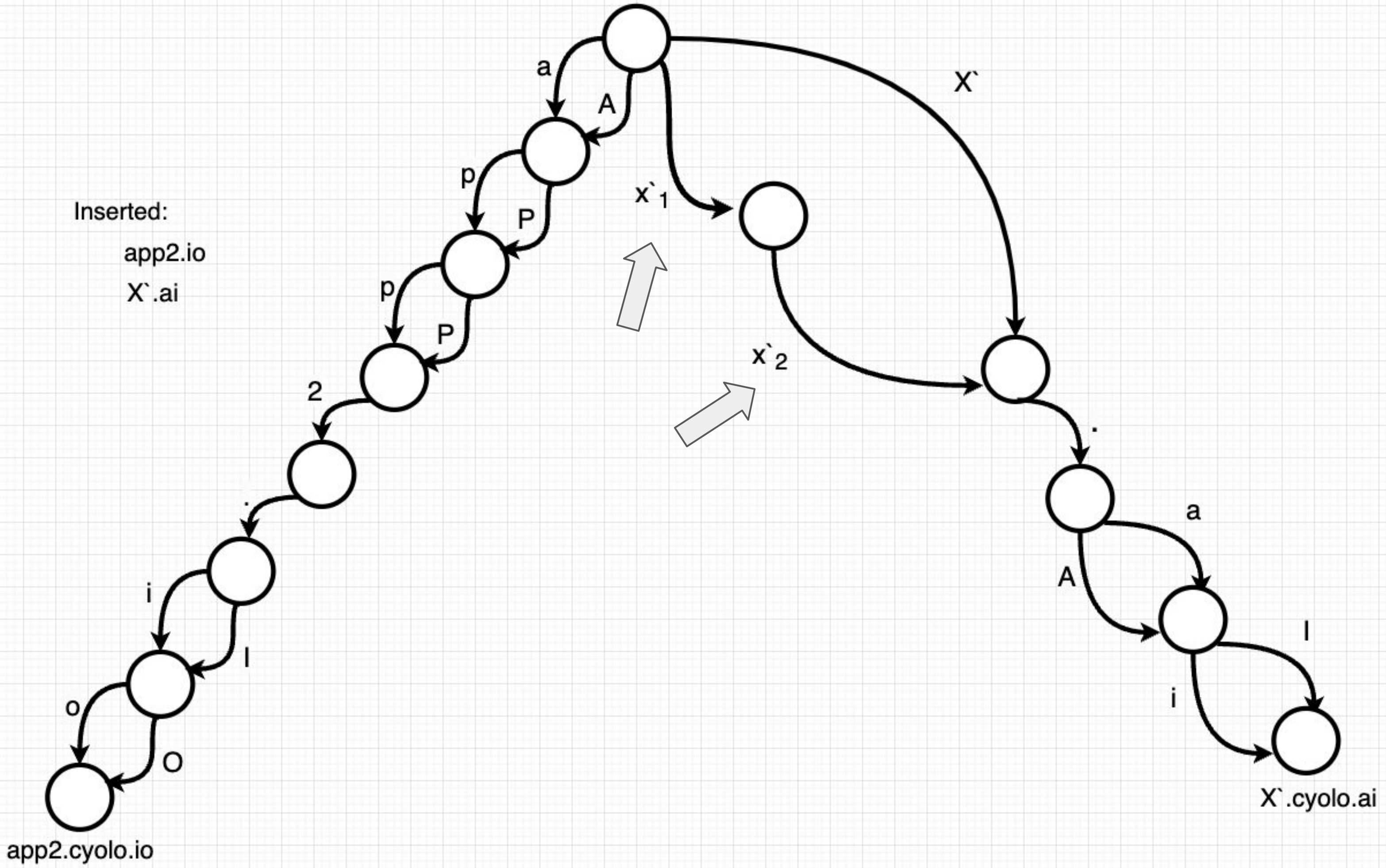
X`.ai



Inserted:

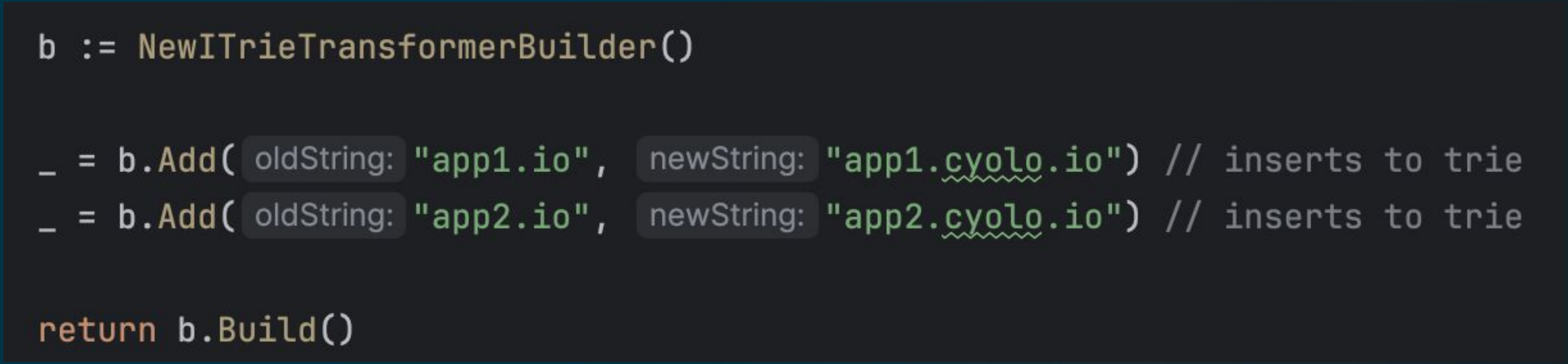
app2.io

X`.ai



The Trie transformer:

Building:



```
b := NewITrieTransformerBuilder()

_ = b.Add( oldString: "app1.io",  newString: "app1.cyolo.io") // inserts to trie
_ = b.Add( oldString: "app2.io",  newString: "app2.cyolo.io") // inserts to trie

return b.Build()
```

The Trie transformer:

Advantages:

The Trie transformer:

Advantages:

- a unique string of size m will have O(m) nodes.

The Trie transformer:

Advantages:

- a unique string of size m will have O(m) nodes.

- one transformer instead of a chain.

The Trie transformer:

Advantages:

- a unique string of size m will have O(m) nodes.

- one transformer instead of a chain.

- rewriting input of size n will take O(n*m) where m is the longest string in the trie.

The Trie transformer:

Case insensitive trie (actually a DAG):

- After shipping the fix and asking for customer’s feedback, the same 20MB JS file that took 30 seconds to process, now takes only **150 ms**.

```
BenchmarkResponseRewriteTransformerForMappings/itrie-with-fb_many-11                  10
151701812 ns/op            151.7 ms/op  128465386 B/op       45 allocs/op
```

The Trie transformer:

Case insensitive trie (actually a DAG):

- After shipping the fix and asking for customer's feedback, the same 20MB JS file that took 30 seconds to process, now takes only **150 ms**.

The Trie transformer:

Case insensitive trie (actually a DAG):

- After shipping the fix and asking for customer's feedback, the same 20MB JS file that took 30 seconds to process, now takes only **150 ms**.

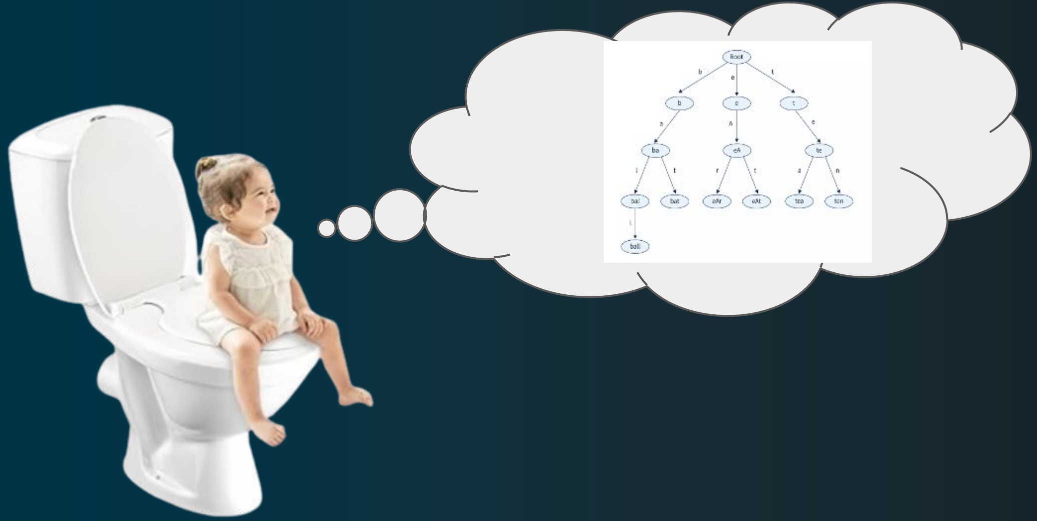
Machtul is back to sleep

Recap:

	Naive chunk-based	Regex	Trie
processing time	6 seconds	30 seconds	150 milliseconds
correctness	Panic	Ok	Ok

Special thanks and credits

to our beloved 10x engineer Elad Shtivi



Thank You!

Questions?