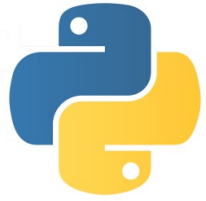


Parsing Contexts

Context Sensitive Grammars for Everyone

Jan Kurš
kurs@iam.unibe.ch
Software Composition Group



Indentation using Petit Parser

```
print "Hello!"
```

```
if (outOf(milk)): ───────────────────────────────────▶ if keyword
```

```
    print "We are out of milk!" ───────────────────▶ block of commands
```

```
    order(milk)
```

```
if (outOf(eggs)): ───────────────────────────────────▶ eggs identifier
```

```
...
```



Indentation using Petit Parser

```
print "Hello!"
```

```
if (outOf(milk)): ───────────────────────────────────▶ if keyword
```

```
    print "We are out of milk!"
```

```
    ───────────────────────────────────▶ block of commands
```

```
    order(milk)
```

```
if (outOf(eggs)): ───────────────────────────────────▶ eggs identifier
```

```
...
```



Indentation using Petit Parser

```
print "Hello!"
```

```
if (outOf(milk)): ───────────────────────────────────▶ if keyword
```

```
    print "We are out of milk!"
```

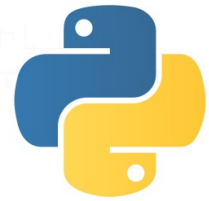
```
    ───────────────────────────────────▶ block of commands
```

```
    order(milk)
```

```
if (outOf(eggs)): ───────────────────────────────────▶ eggs identifier
```

```
...
```

```
ifKeyword := 'if' asParser.
```



Indentation using Petit Parser

```
print "Hello!"
```

```
if (outOf(milk)): ───────────────────────────────────▶ if keyword
```

```
    print "We are out of milk!" ───────────────────▶ block of commands
```

```
    order(milk)
```

```
if (outOf(eggs)): ───────────────────────────────────▶ eggs identifier
```

```
...
```



Indentation using Petit Parser

```
print "Hello!"
```

```
if (outOf(milk)): ───────────────────────────────────▶ if keyword
```

```
    print "We are out of milk!"
```

```
    ───────────────────────────────────▶ block of commands
```

```
    order(milk)
```

```
if (outOf(eggs)): ───────────────────────────────────▶ eggs identifier
```

...

```
identifier := #letter, (#letter / #digit) star.
```



Indentation using Petit Parser

```
print "Hello!"
```

```
if (outOf(milk)): ───────────────────────────────────▶ if keyword
```

```
    print "We are out of milk!" ───────────────────▶ block of commands  
    order(milk)
```

```
if (outOf(eggs)): ───────────────────────────────────▶ eggs identifier
```

```
...
```



Indentation using Petit Parser

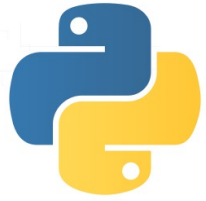
```
print "Hello!"
```

```
if (outOf(milk)): └──────────────────────────────────────────▶ if keyword  
    print "We are out of milk!" └──────────────────────────────────────────▶ block of commands  
    order(milk)
```

```
if (outOf(eggs)): └──────────────────────────────────────────▶ eggs identifier
```

...

```
block := indent, command plus, dedent.
```

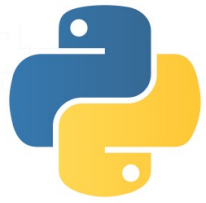



Indentation using Petit Parser

```
print "We are out of milk!"  
order(milk)
```

→ block of commands

`block := indent, command plus, dedent.`



Indentation using Petit Parser

```
print "We are out of milk!"  
order(milk)
```

→ block of commands

`block := indent, command plus, dedent.`

`indent := ???`

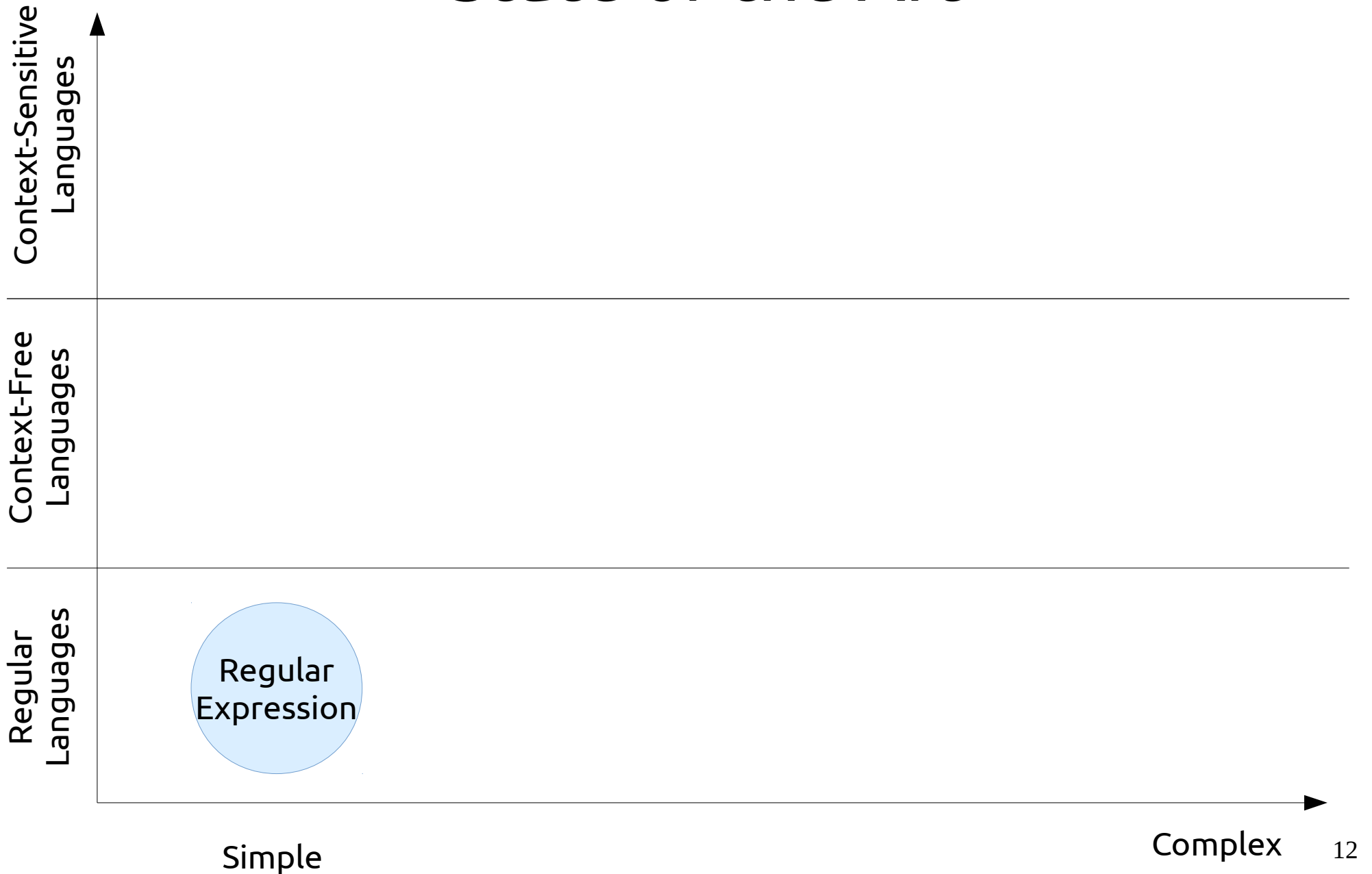
`dedent := ???`



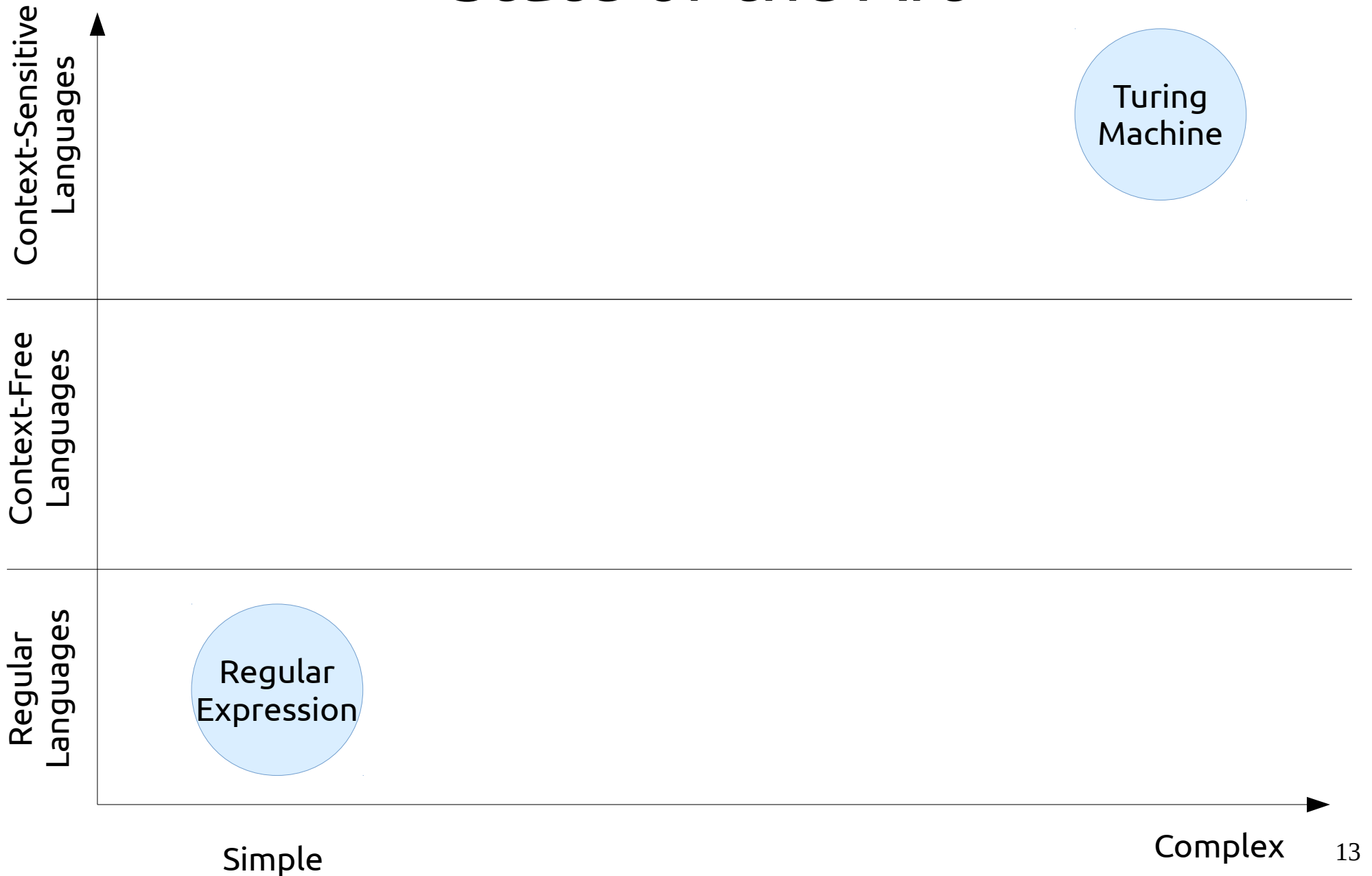
State of the Art



State of the Art



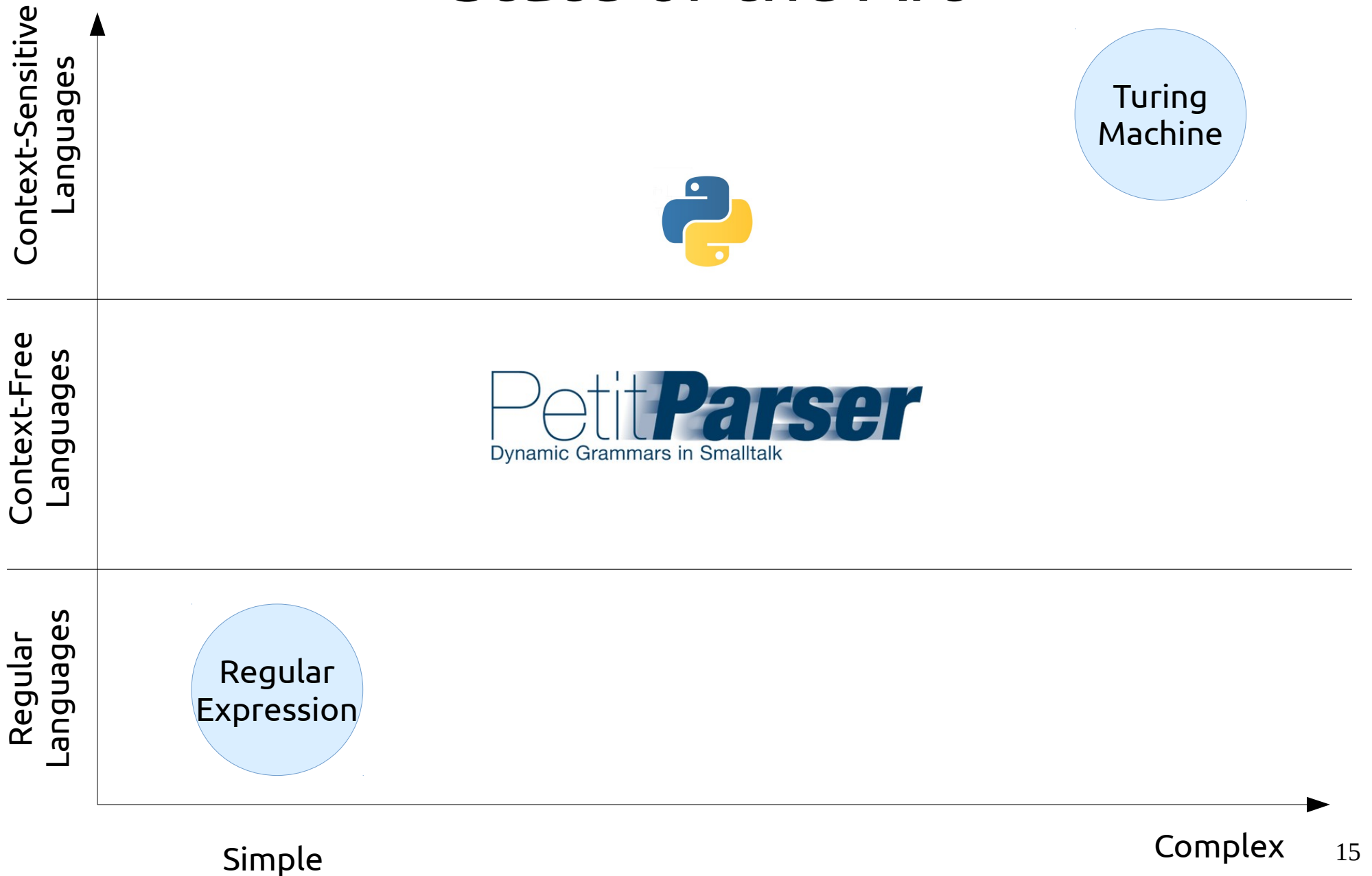
State of the Art



State of the Art



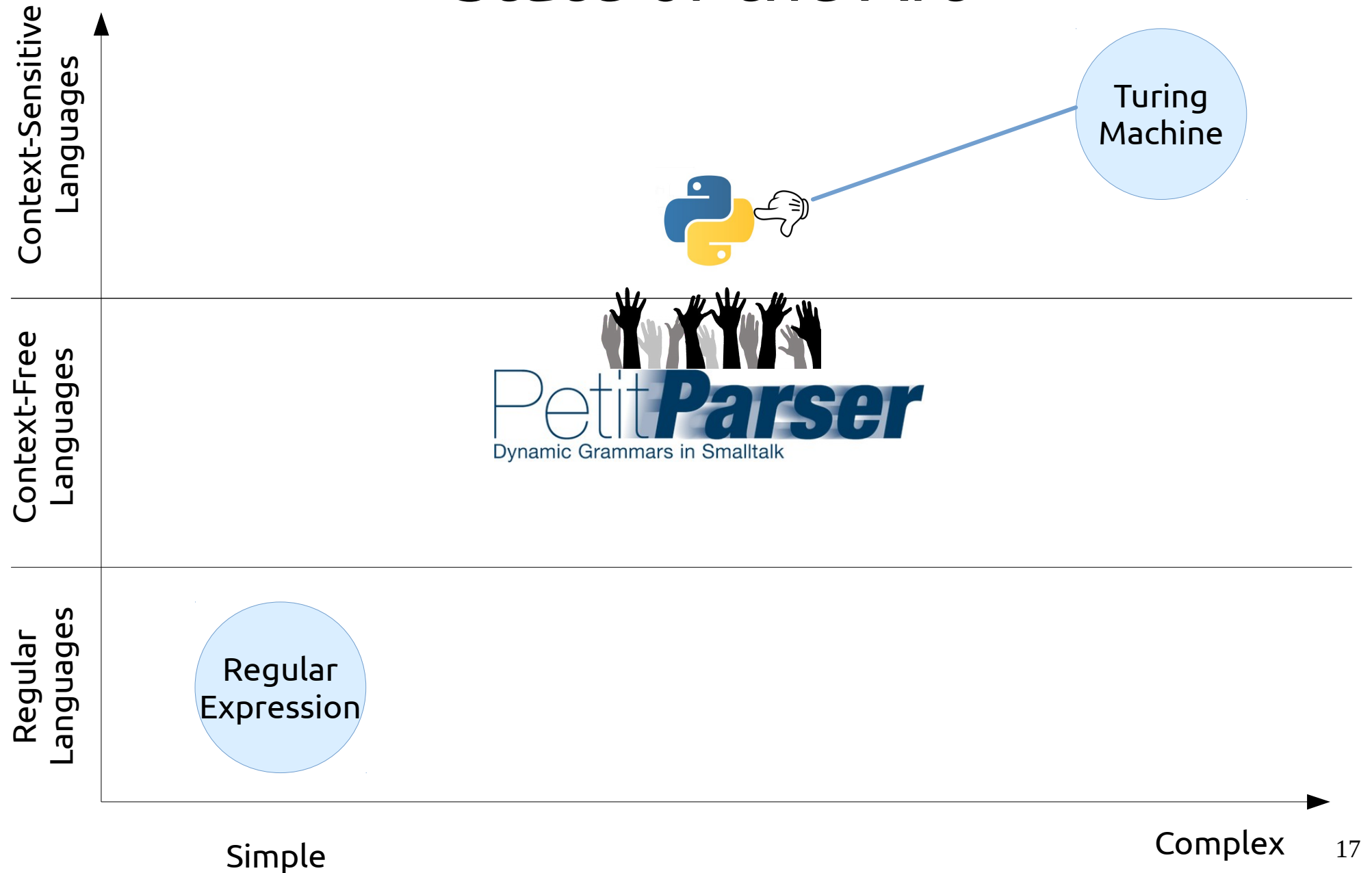
State of the Art



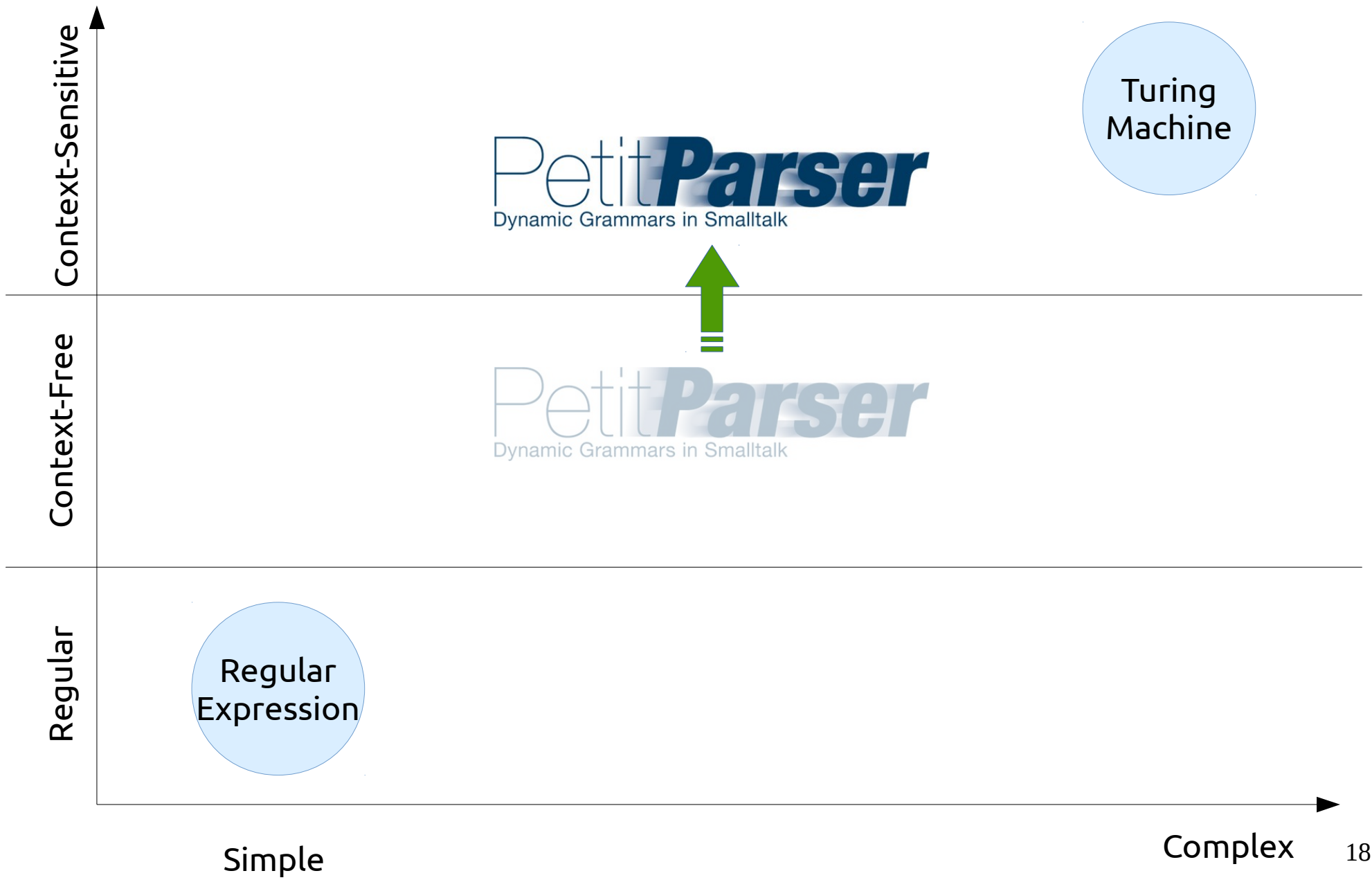
State of the Art



State of the Art



Goal



Regular Expression

Turing Machine

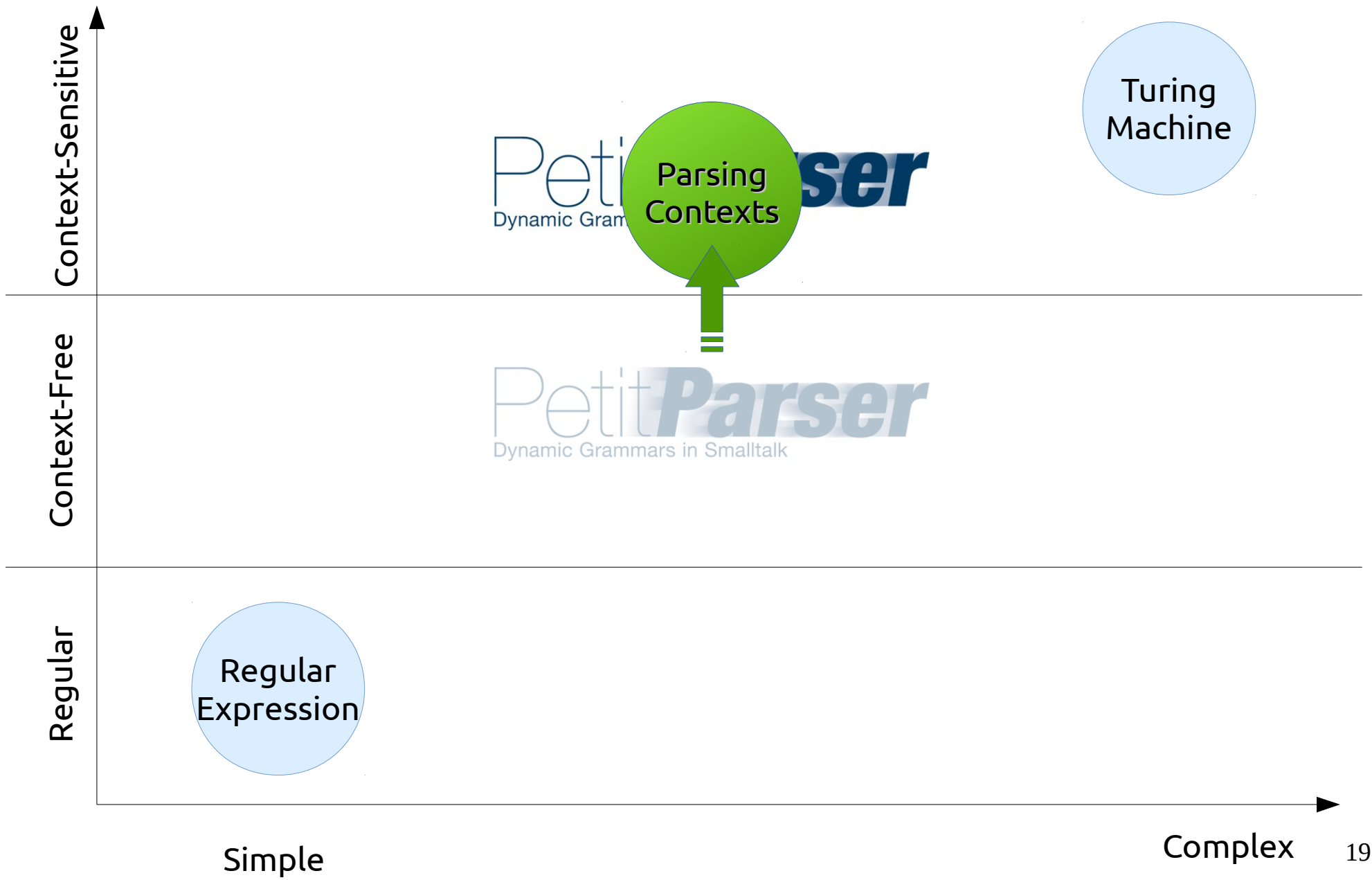
Petit **Parser**
Dynamic Grammars in Smalltalk

Petit *Parser*
Dynamic Grammars in Smalltalk

Simple

Complex

Goal



Turing Machine

Petit Parser
Dynamic Grammars

Parsing Contexts

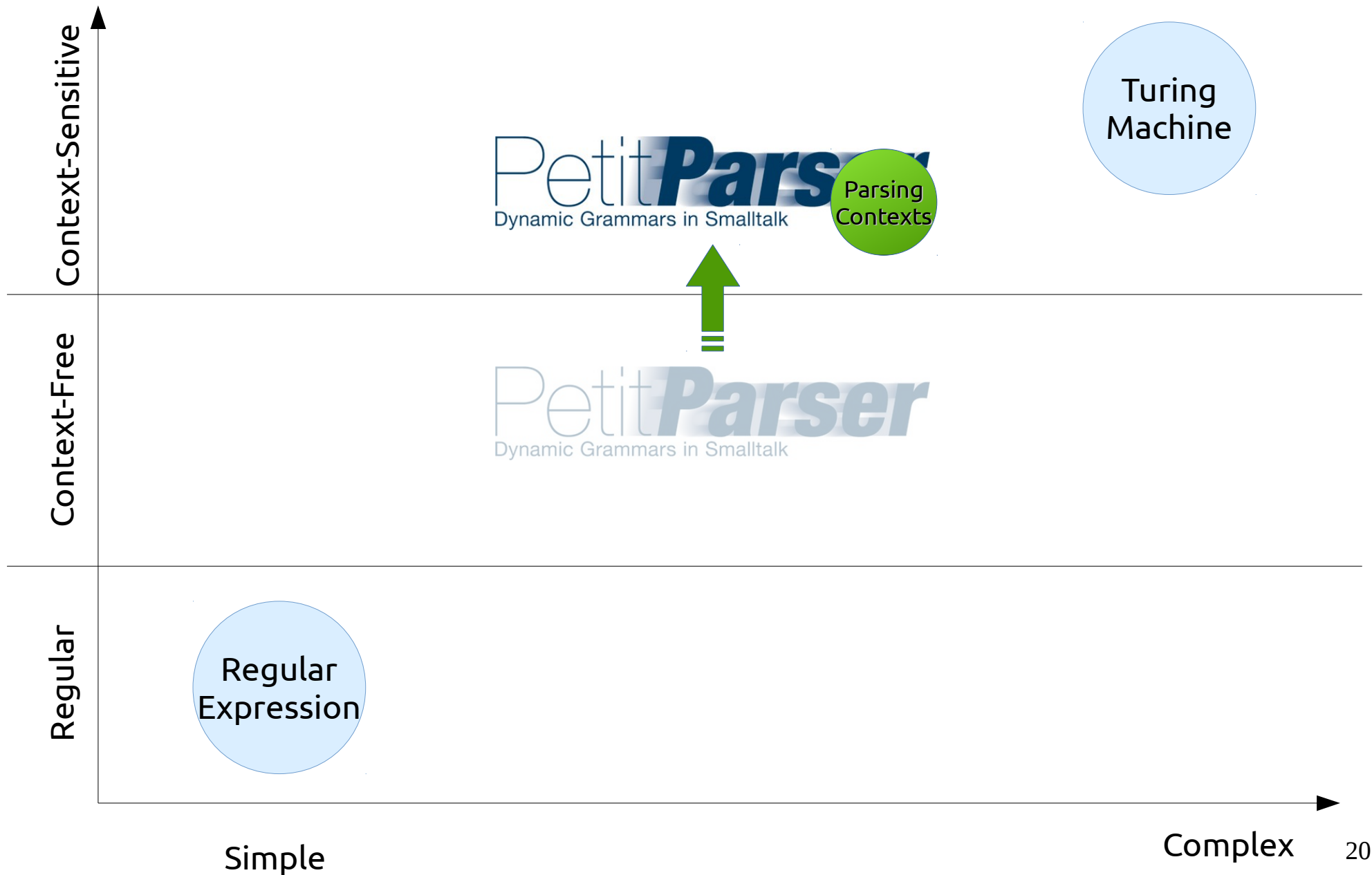
Petit Parser
Dynamic Grammars in Smalltalk

Regular Expression

Simple

Complex

Goal

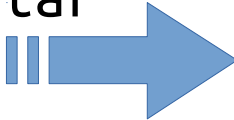


PetitParser uses **Parser Combinators**

PetitParser uses **Parser Combinators**

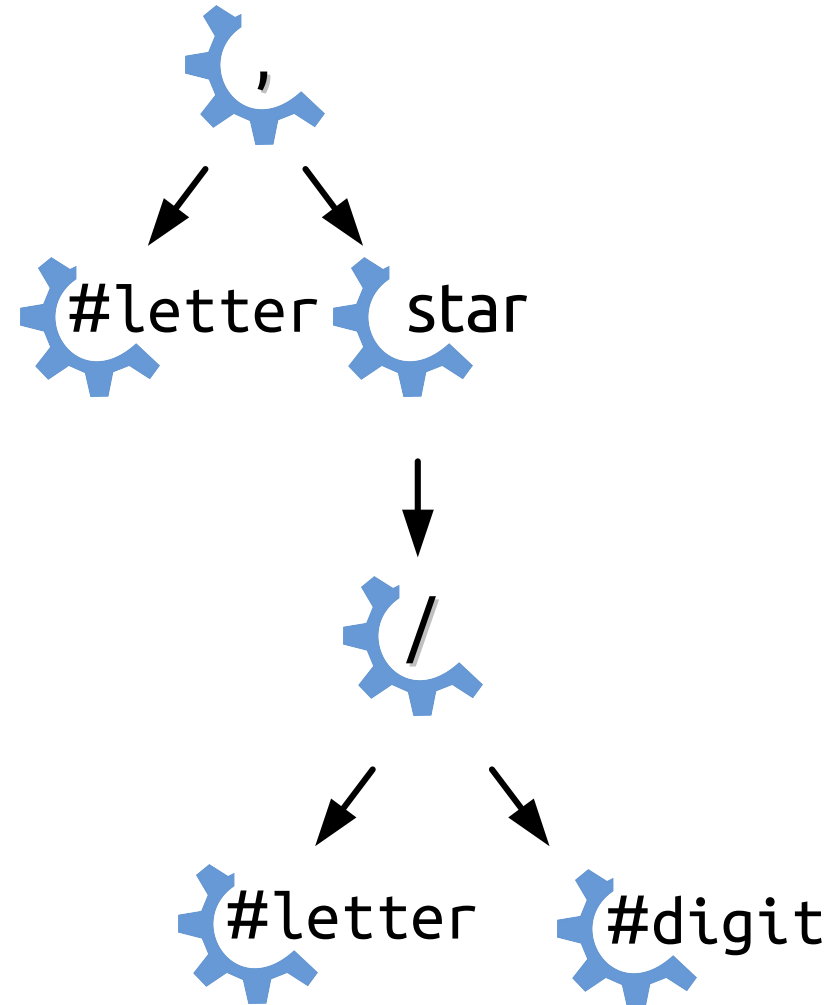
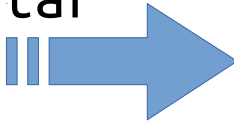
`#letter, (#letter / #digit) star`

PetitParser uses **Parser Combinators**

`#letter, (#letter / #digit) star` 

PetitParser uses **Parser Combinators**

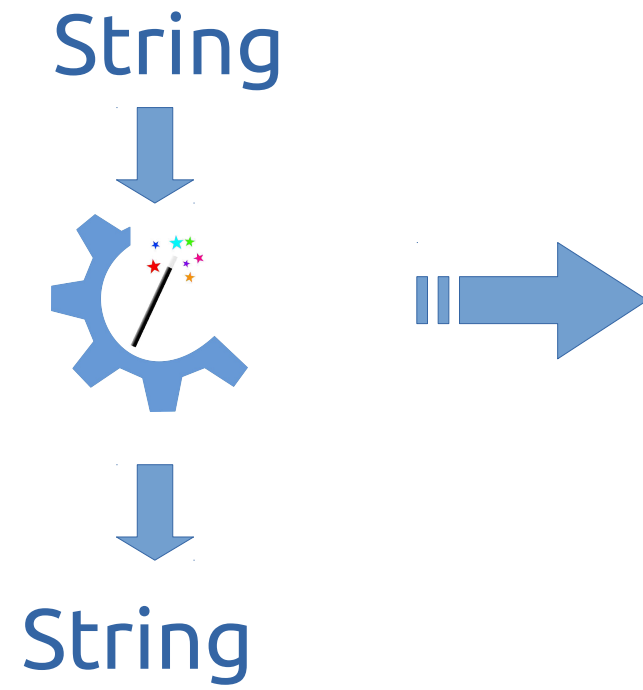
#letter, (#letter / #digit) star

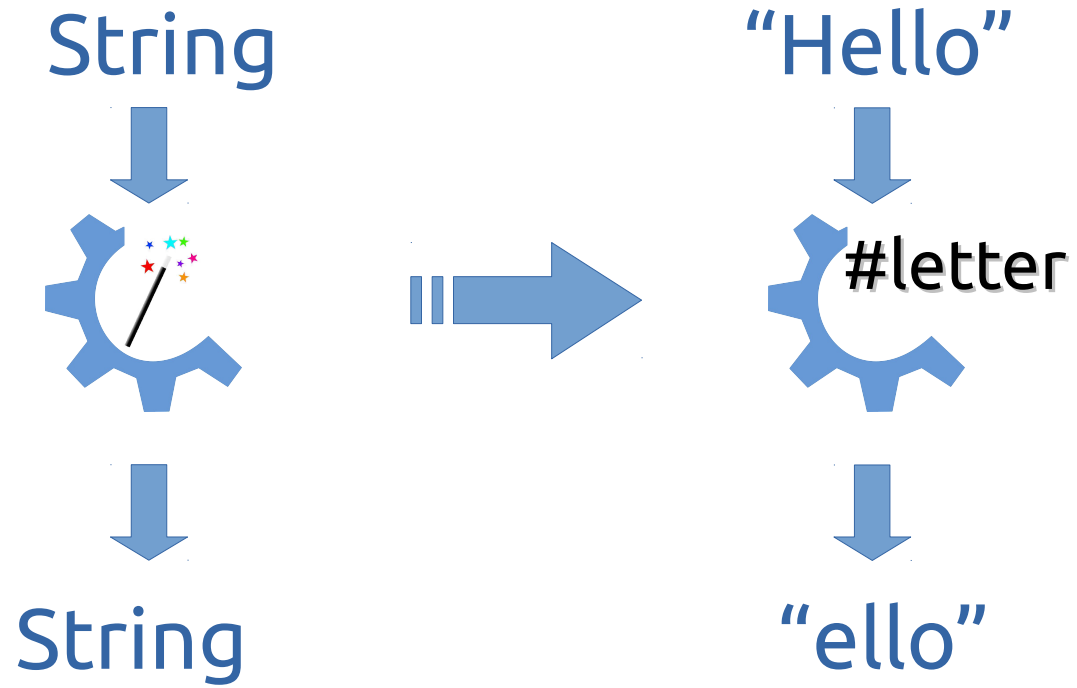


String



String







Indentation Example

```
print "We are out of milk!"  
order(milk)
```

→ block of commands

block := indent, command plus, dedent.



Indentation Example

```
print "We are out of milk!"  
order(milk)
```

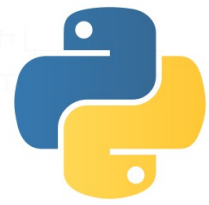
→ block of commands

block := indent, command plus, dedent.

indent := ???

dedent := ???





Indentation Example



```
print "We are out of milk!"  
order(milk)
```

→ block of commands

`block := indent, command plus, dedent.`



Indentation Example



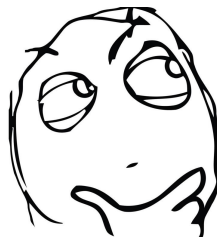
```
print "We are out of milk!"  
order(milk)
```

→ block of commands

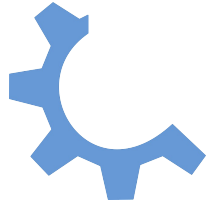
block := indent, command plus, dedent.

indent := ???

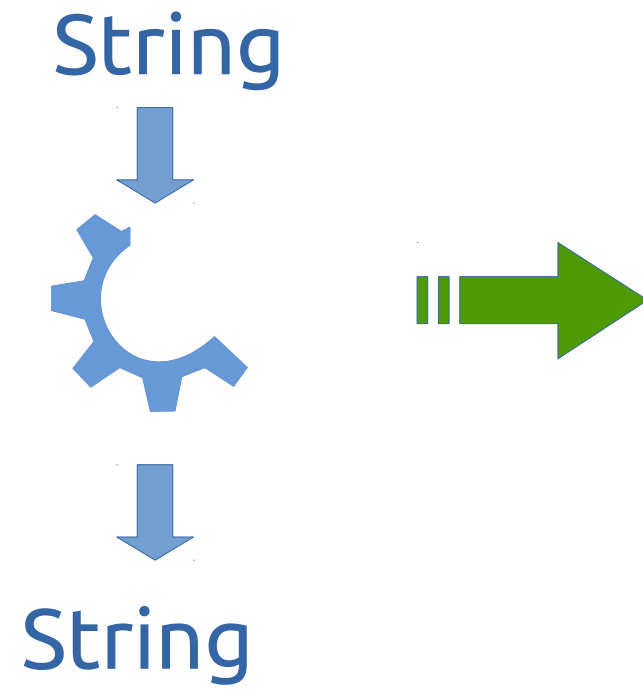
dedent := ???

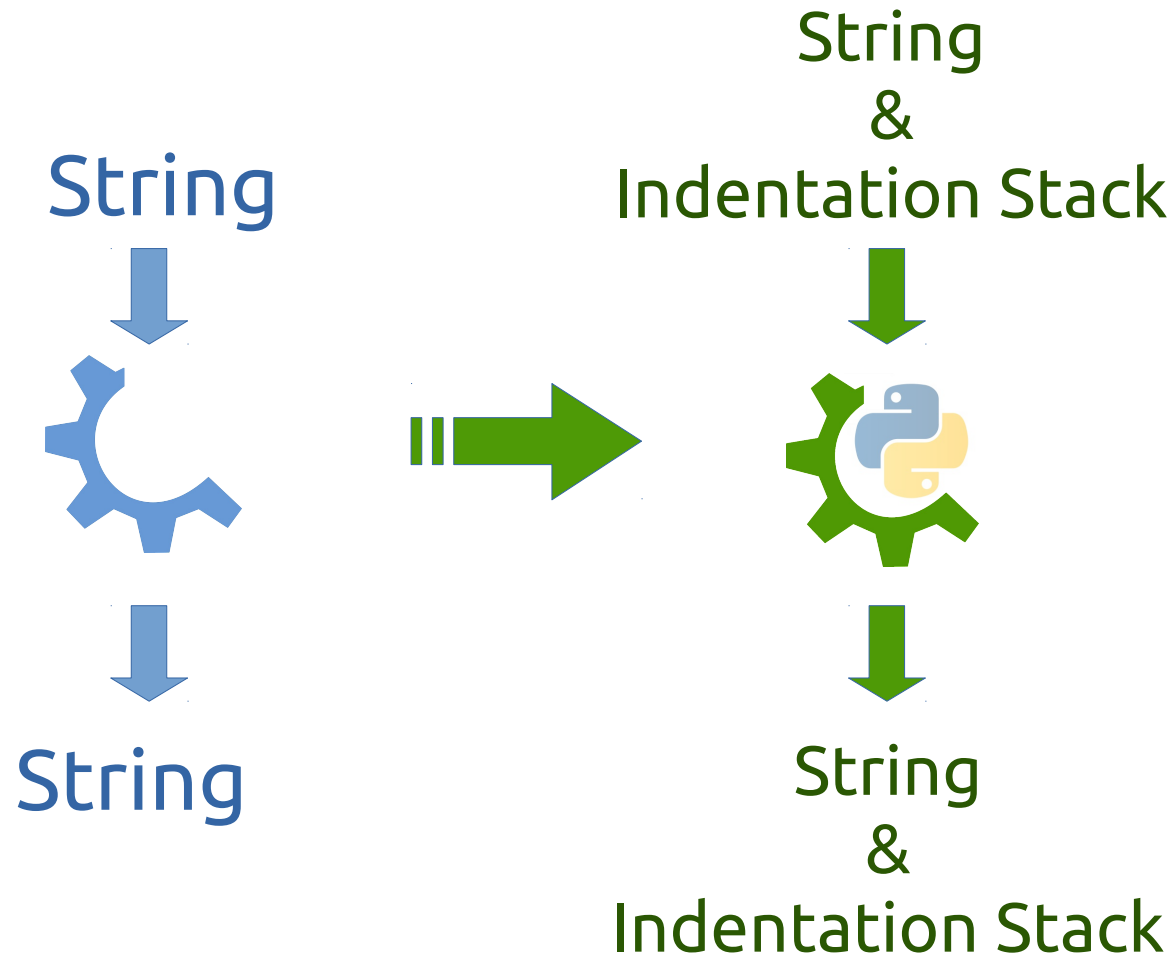


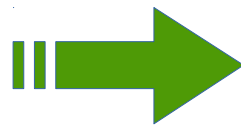
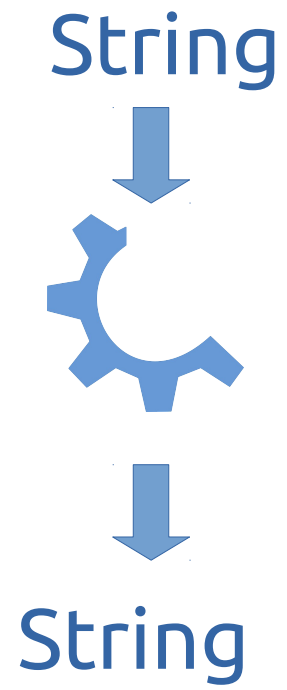
String

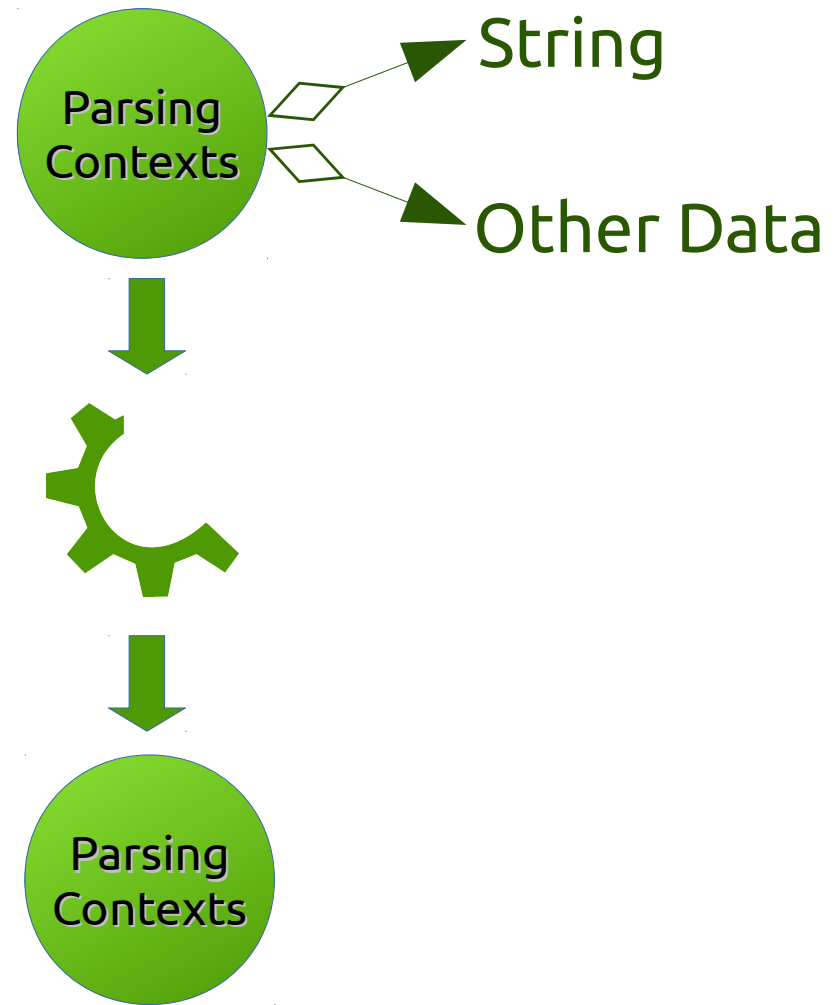
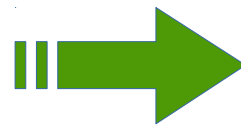
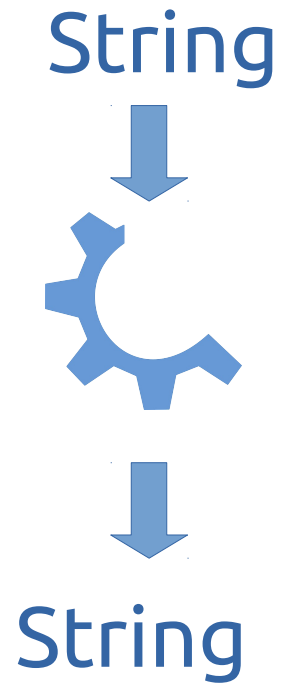


String







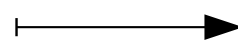




Indentation Example



print "We are out of milk!"
order(milk)



block of commands

block :=

```
[context |
 | column indentation stack |

(context isBeginOfLine) iffFalse: [
 ^ Failure
].
context consumeLeadingWhitespace.

" Save the current column "
column := context stream column.
stack := (context propertyAt: #indent).
indentation := stack top.

(column > indentation) iffTrue: [
 stack push: column.
 ^ #indent
].
^ Failure
]
```

, command plus,

```
[context |
 | column referenceIndentation stack |

(context isBeginOfLine) iffFalse: [
 ^ Failure
].
context consumeLeadingWhitespace.
column := context stream column.

" Restore previous column from the "
" stack and compare with current "
stack := (context propertyAt: #indent).
stack pop.
referenceIndentation := stack top.

(column == referenceIndentation) iffTrue: [
 ^ #dedent
].
^ Failure
]
```





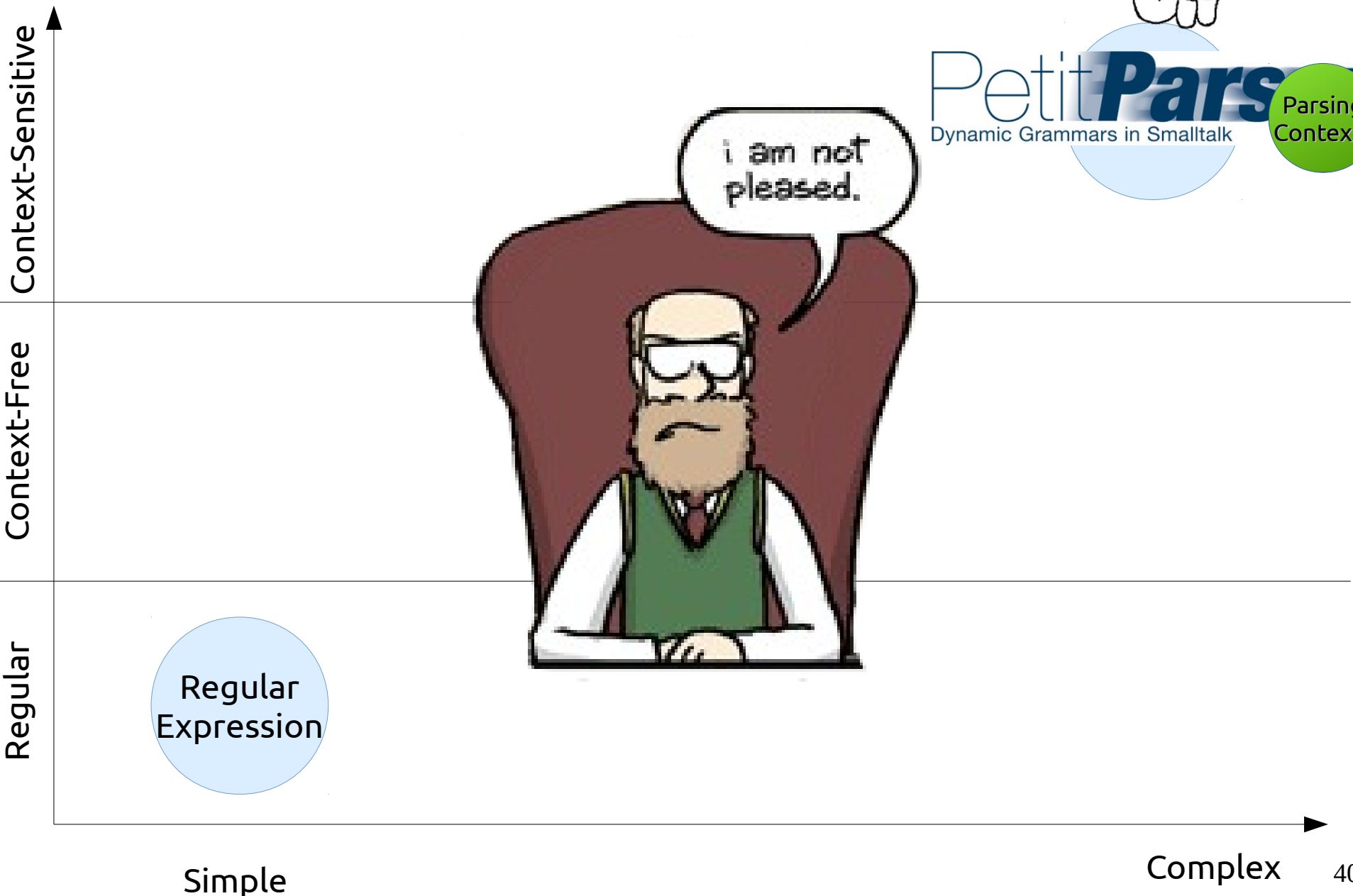
Petit **Pars**
Dynamic Grammars in Smalltalk

Parsing
Contexts





PetitPars
Dynamic Grammars in Smalltalk
Parsing Contexts





Grammar Implementors



Grammar
Implementors



Framework
Implementors



$$N \leftarrow (N \cup T)^*$$

Grammar
Implementors



Framework
Implementors



Grammar
Implementors

$$N \leftarrow (N \cup T)^*$$

indent, command plus, dedent



Framework
Implementors



Grammar Implementors

$$N \leftarrow (N \cup T)^*$$

indent, command plus, dedent



Framework Implementors

```

[:context |
 | column indentation stack |

(context isBeginOfLine) ifFalse: [
 ^ Failure
].
context consumeLeadingWhitespace.

" Save the current column "
column := context stream column.
stack := (context propertyAt: #indent).
indentation := stack top.

(column > indentation) ifTrue: [
 stack push: column.
 ^ #indent
].
^ Failure
]

```

, command plus,

```

[:context |
 | column referenceIndentation stack |

(context isBeginOfLine) ifFalse: [
 ^ Failure
].
context consumeLeadingWhitespace.
column := context stream column.

" Restore previous column from the "
" stack and compare with current "
stack := (context propertyAt: #indent).
stack pop.
referenceIndentation := stack top.

(column == referenceIndentation) ifTrue: [
 ^ #dedent
].
^ Failure
]

```



Context-Sensitive

Context-Free

Regular

Regular
Expression

Simple

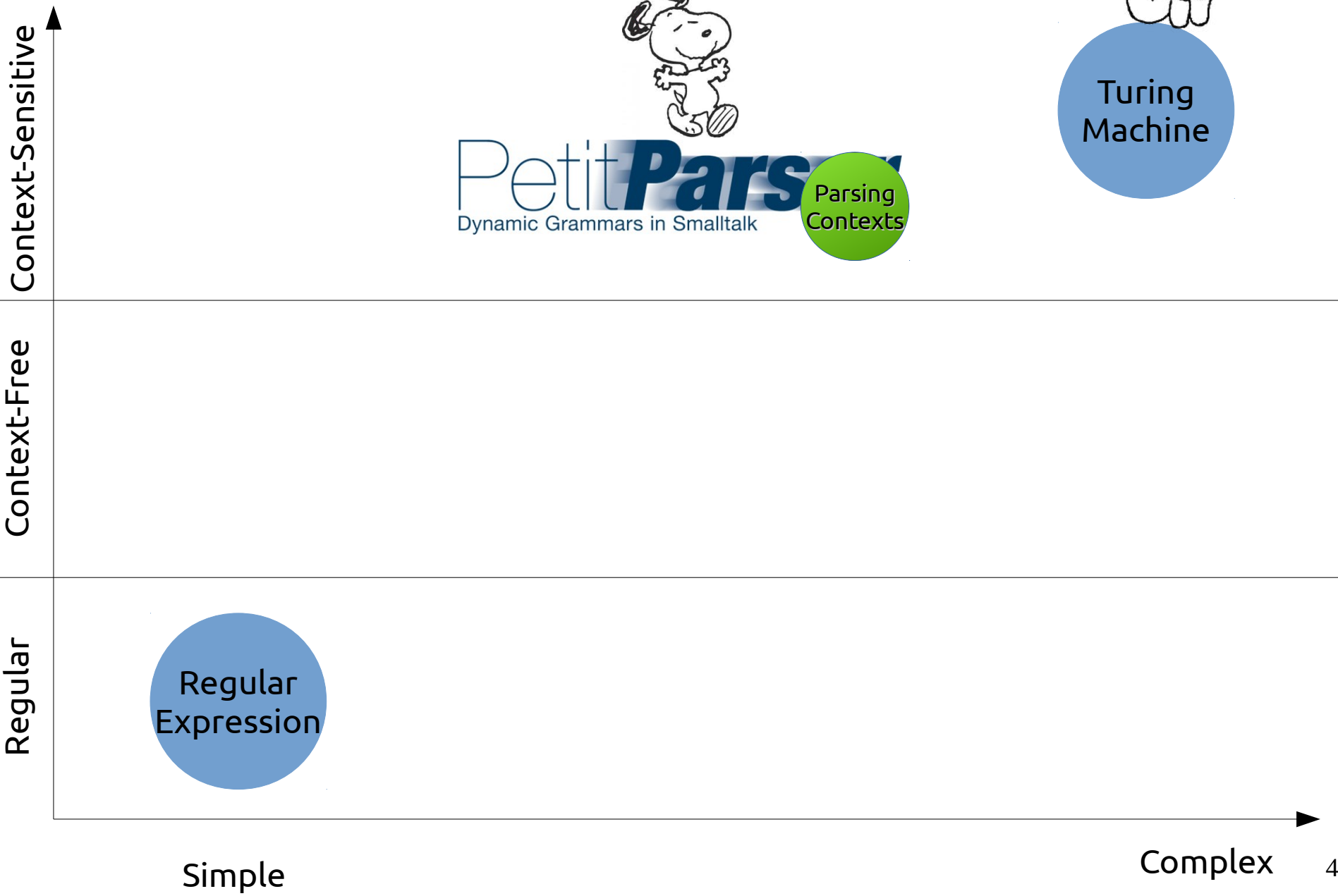
Petit *Pars*
Dynamic Grammars in Smalltalk

Parsing
Contexts

Turing
Machine



Complex



Petit **Parser**
Dynamic Grammars in Smalltalk

Petit **Parser**
Dynamic Grammars in Smalltalk

Parsing
Contexts

PetitParser
Dynamic Grammars in Smalltalk

PetitParser
Dynamic Grammars in Smalltalk
Parsing
Contexts



indentation





indentation



tolerant parsing





indentation



tolerant parsing



debugging



http://smalltalkhub.com/#!/~JanKurs/PetitParser



[Home](#) [Explore](#)

JanKurs

JanKurs / PetitParser ✓ PUBLIC

0

[Overview](#) [Source](#) [Commits](#) [Contributors](#) [Watchers](#) [Settings](#)

Project Infos

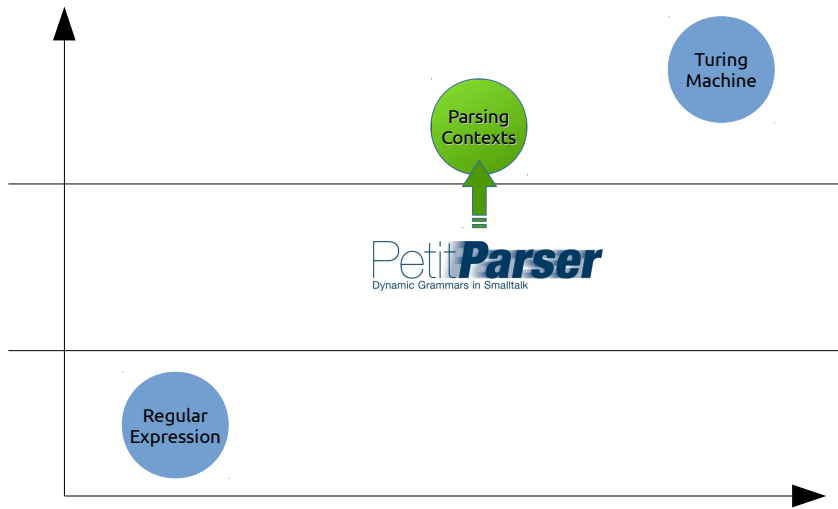
Edit infos

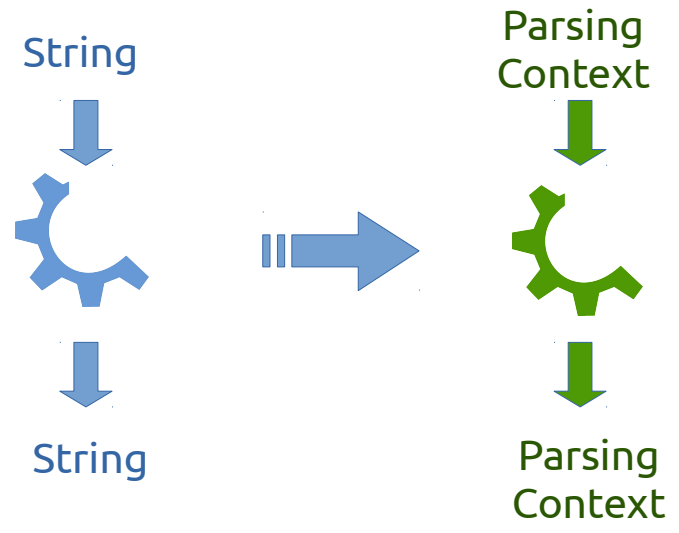
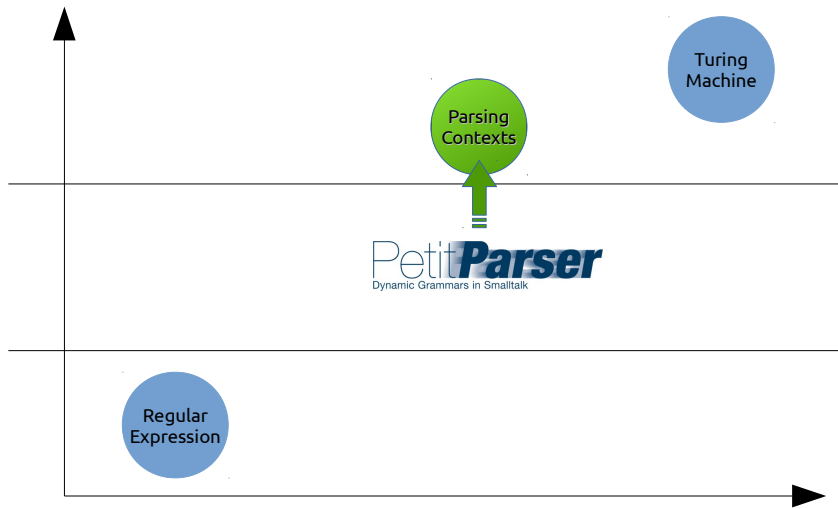
Monticello registration

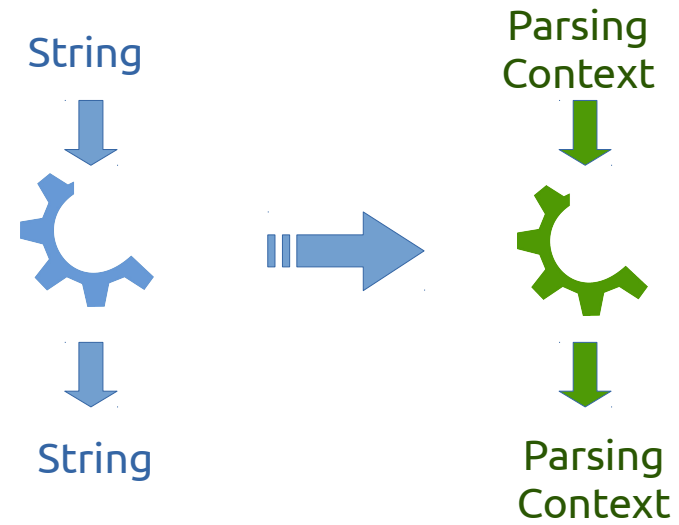
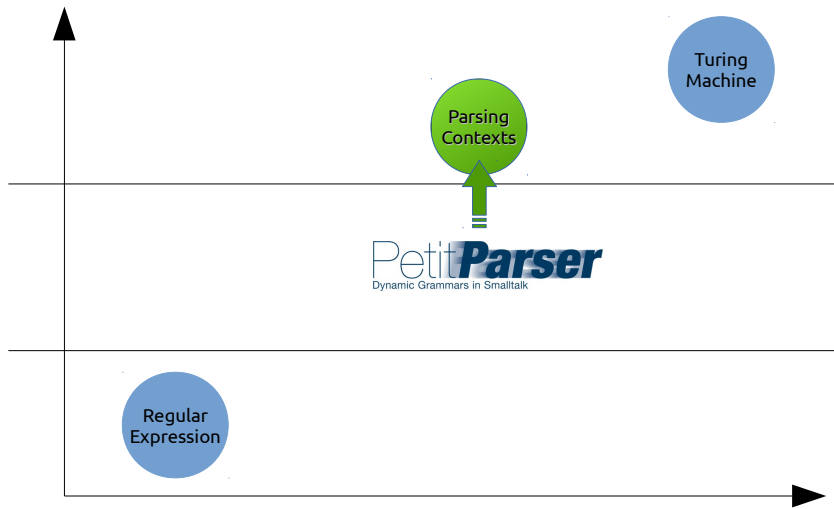
License	MIT
Tags	
Creation date	Fri Apr 05 2013
Website	

```
MCHttpRepository
location: 'http://smalltalkhub.com/mc/JanKurs/PetitParser/main'
user: ""
password: ""
```









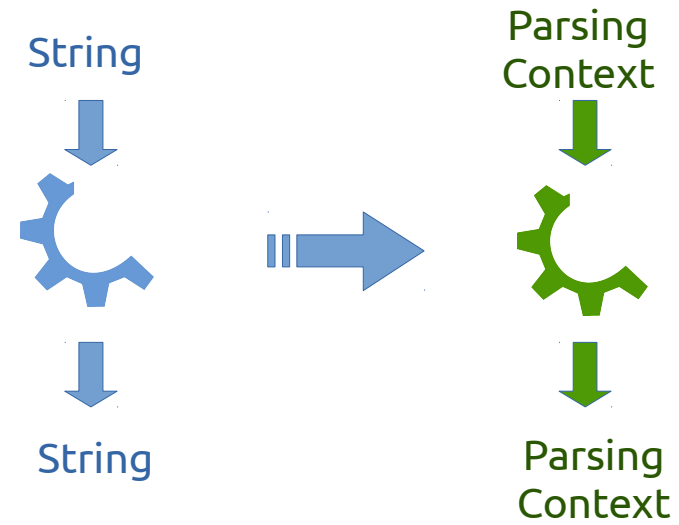
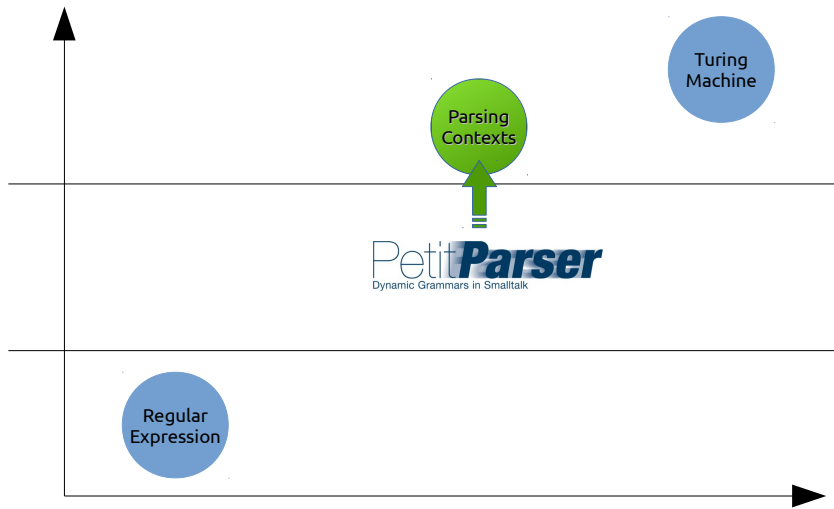
$$N \leftarrow (N \cup T)^*$$



Grammar Implementor



Framework Implementors






$$N \leftarrow (N \cup T)^*$$



Grammar Implementor



Framework Implementors

-  indentation
-  tolerant parsing
-  debugging

