UNIVERSITÄT
BERN

Parsing Contexts

for PetitParser

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

Indentation Example

```
print “Hello!”
if (outOf(milk)):
    print “We are out of milk!”
    order(milk)
if (outOf(eggs)):
    …
```

if keyword

block of commands

eggs identifier

Indentation Example

```
print “Hello!”
if (outOf(milk)):
    print “We are out of milk!”
    order(milk)
if (outOf(eggs)):
    …
```

if keyword

block of commands

eggs identifier

Indentation Example

```
print “Hello!”
if (outOf(milk)):
    print “We are out of milk!”
    order(milk)
if (outOf(eggs)):
    …
```

if keyword

block of commands

eggs identifier

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

Indentation Example

```
print “Hello!”
if (outOf(milk)):
    print “We are out of milk!”
    order(milk)
if (outOf(eggs)):
    …
```

if keyword

block of commands

eggs identifier

Indentation Example

```
print “Hello!”
if (outOf(milk)):
    print “We are out of milk!”
    order(milk)
if (outOf(eggs)):
    ...
```

if keyword

block of commands

eggs identifier

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

Indentation Example

```
print “Hello!”
if (outOf(milk)):
    print “We are out of milk!”
    order(milk)
if (outOf(eggs)):
    …
```

if keyword

block of commands

eggs identifier

Indentation Example

```
print “Hello!”
if (outOf(milk)):
    print “We are out of milk!”
    order(milk)
if (outOf(eggs)):
    …
```

if keyword

block of commands

eggs identifier

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

Indentation Example

print “We are out of milk!”
order(milk)

block of commands

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

Indentation Example

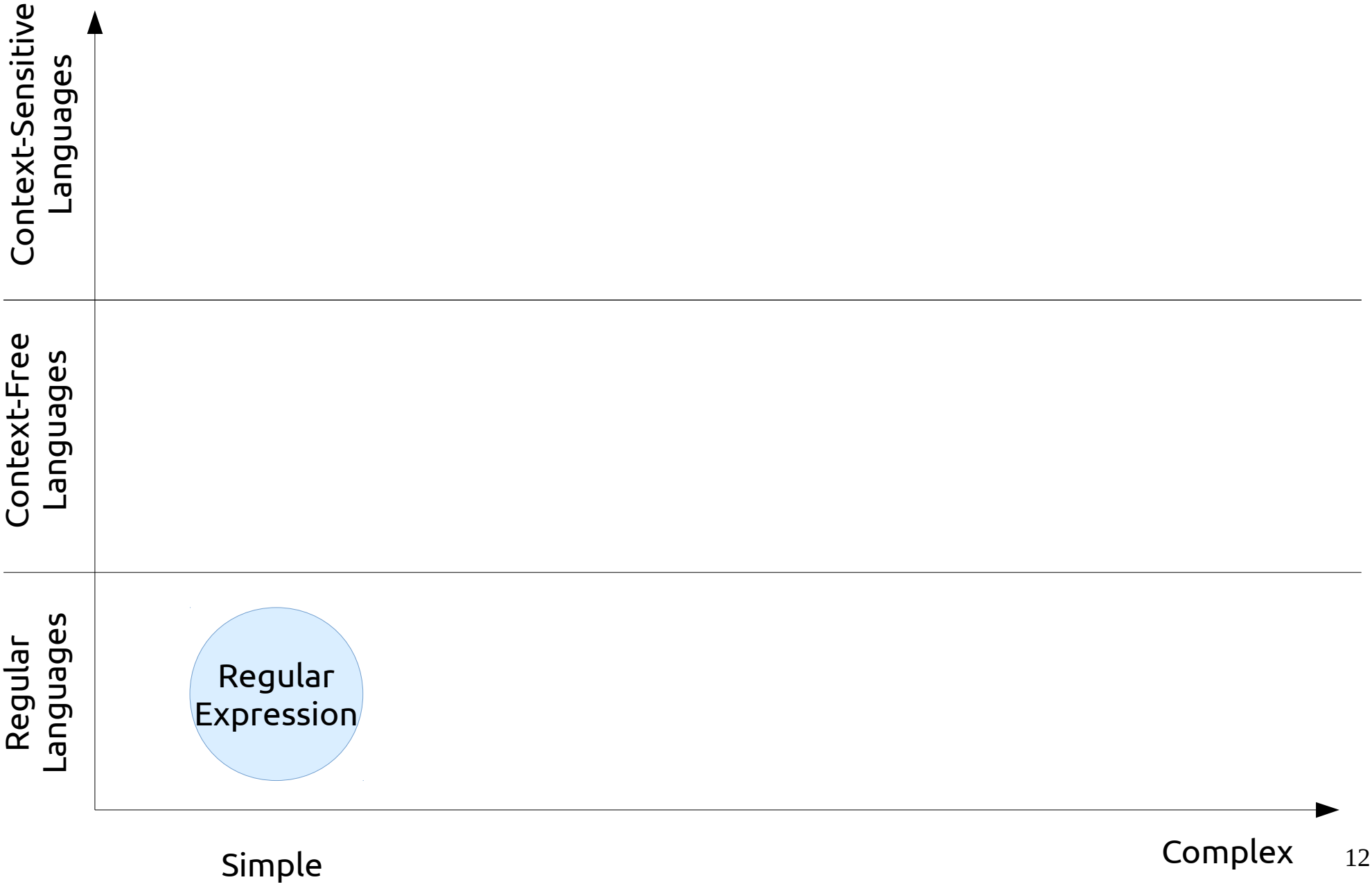
print “We are out of milk!”
order(milk)

→ block of commands

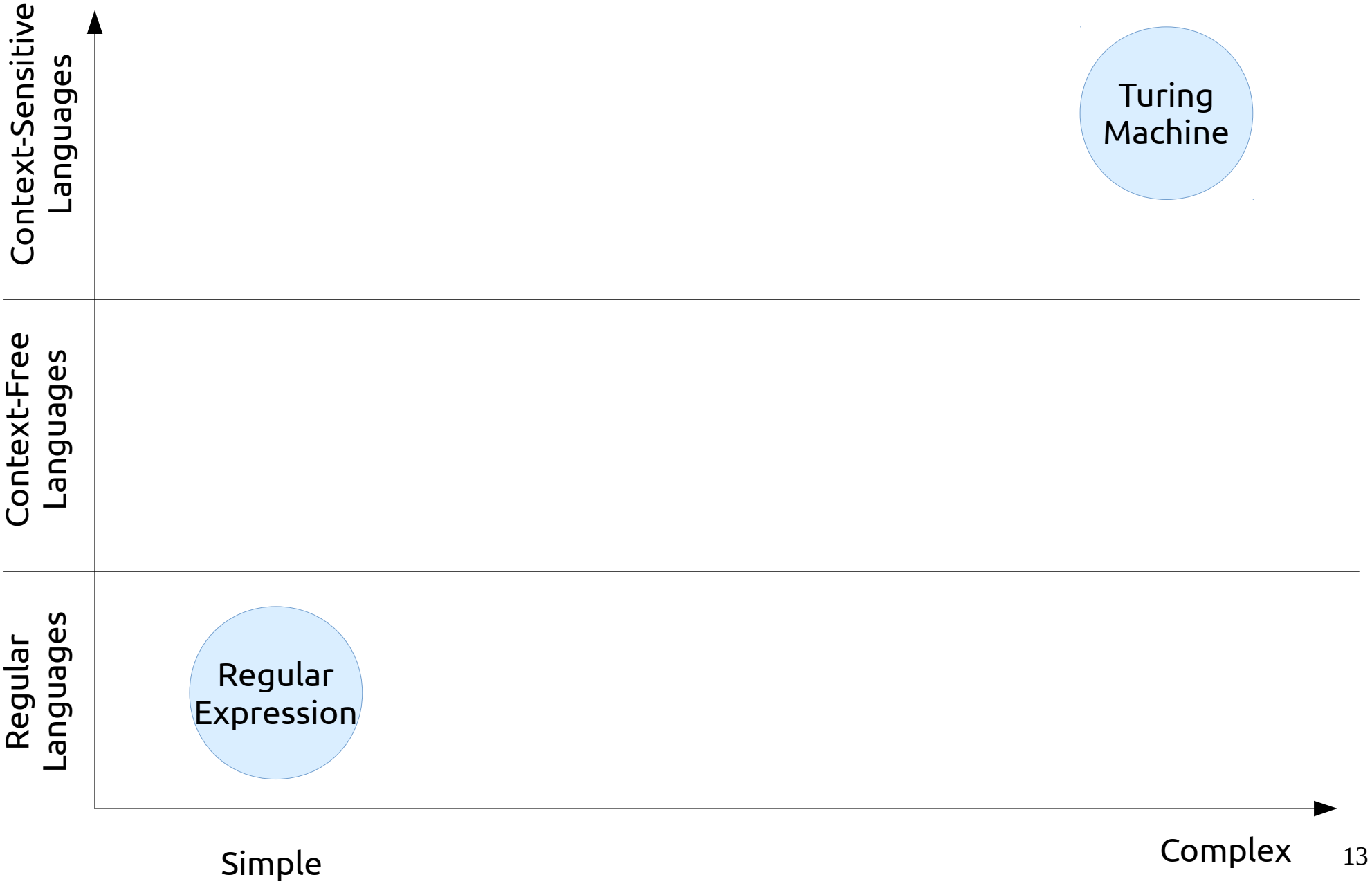
```
block  := indent, command plus, dedent.

indent := ???
dedent := ???
```

State of the Art

State of the Art

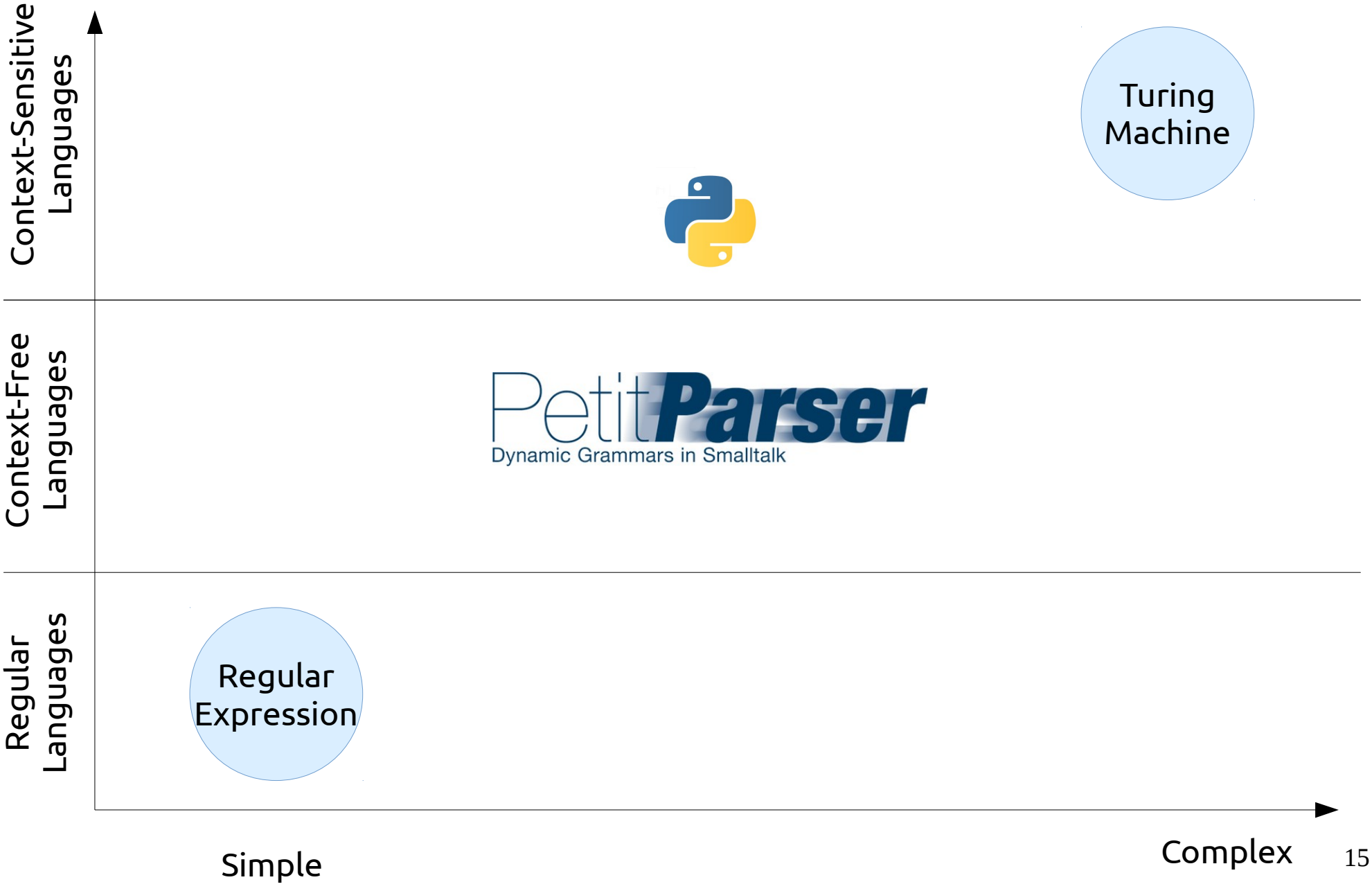


State of the Art

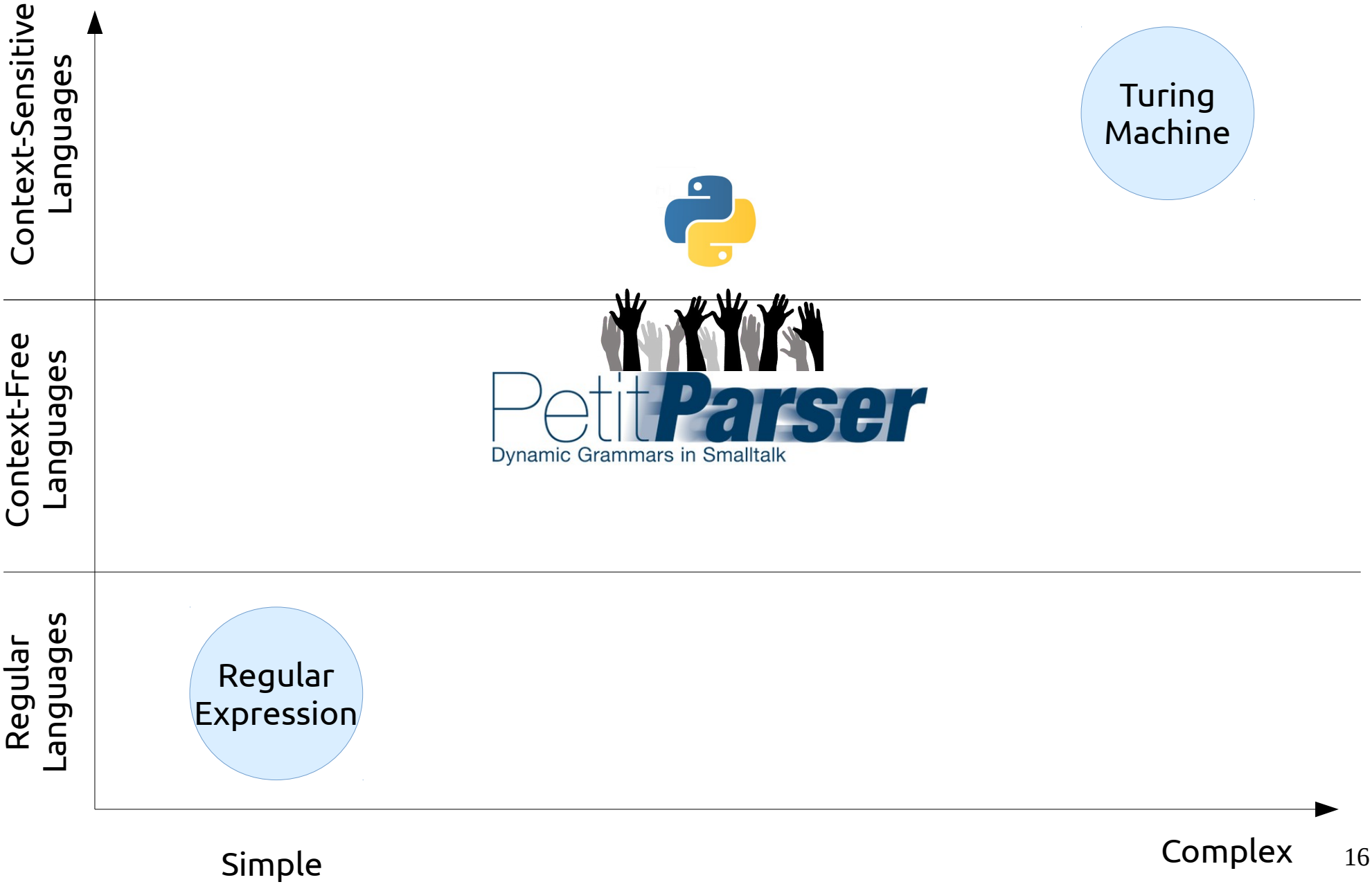


State of the Art

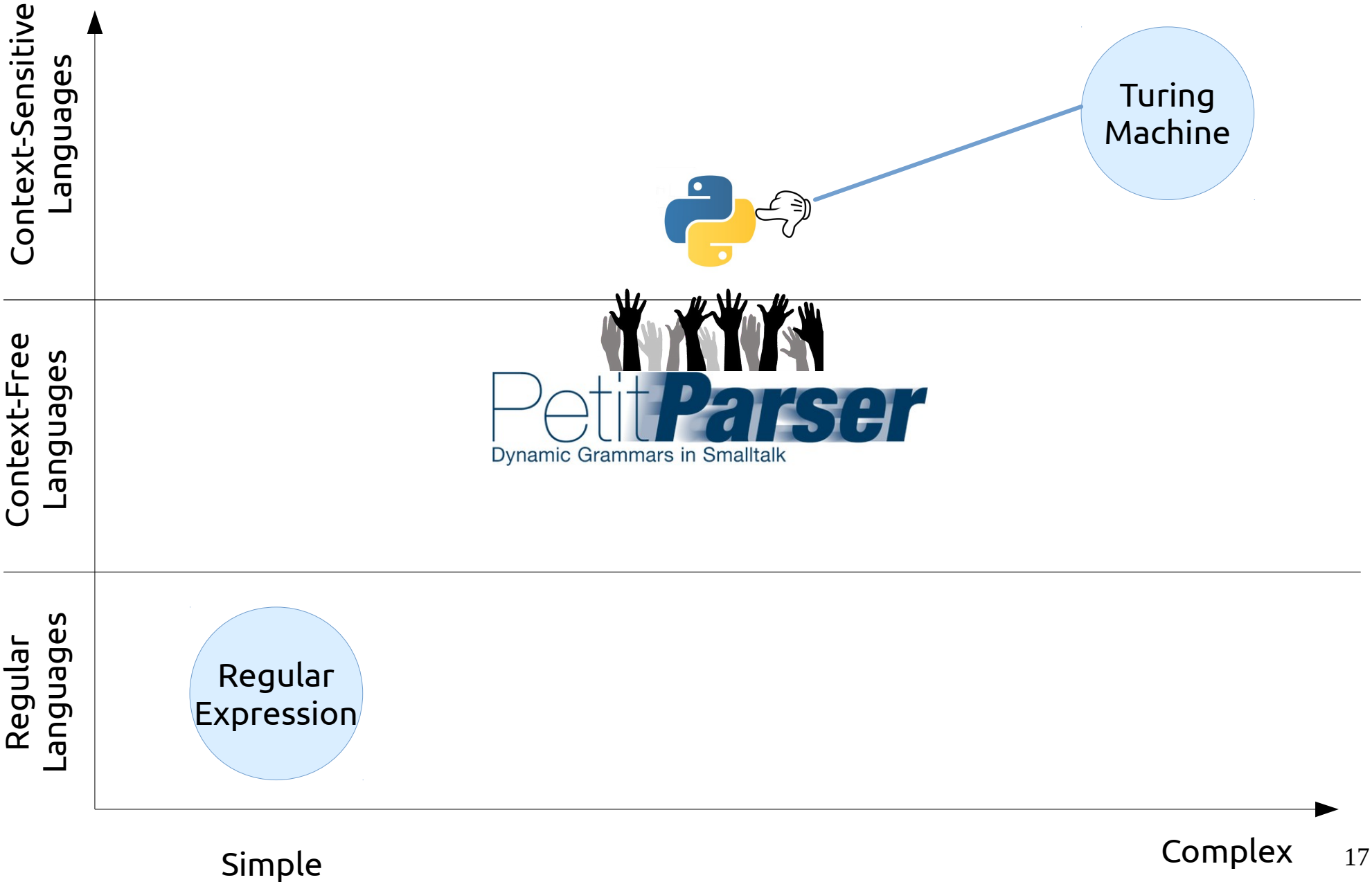
State of the Art



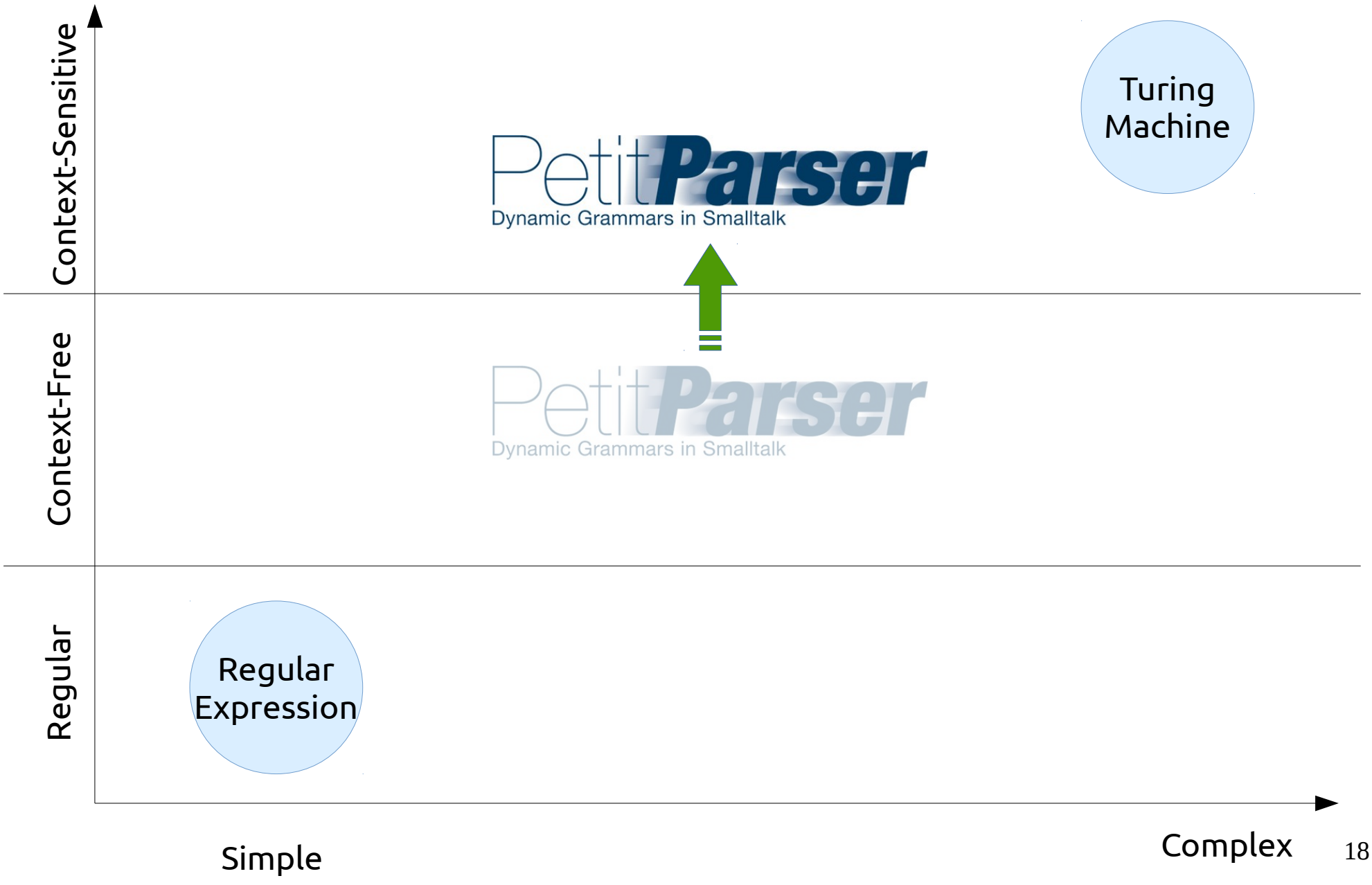
State of the Art



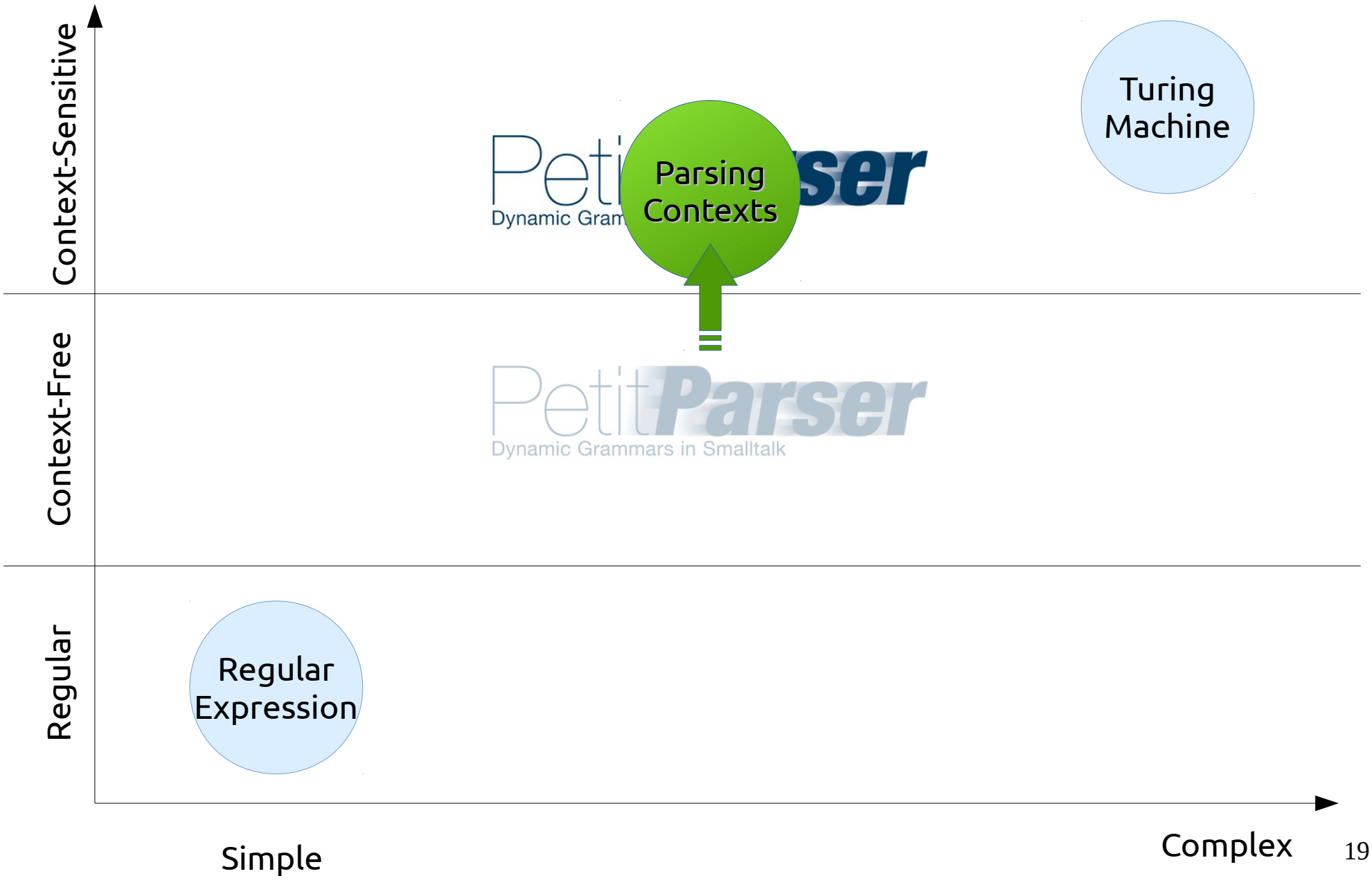
State of the Art



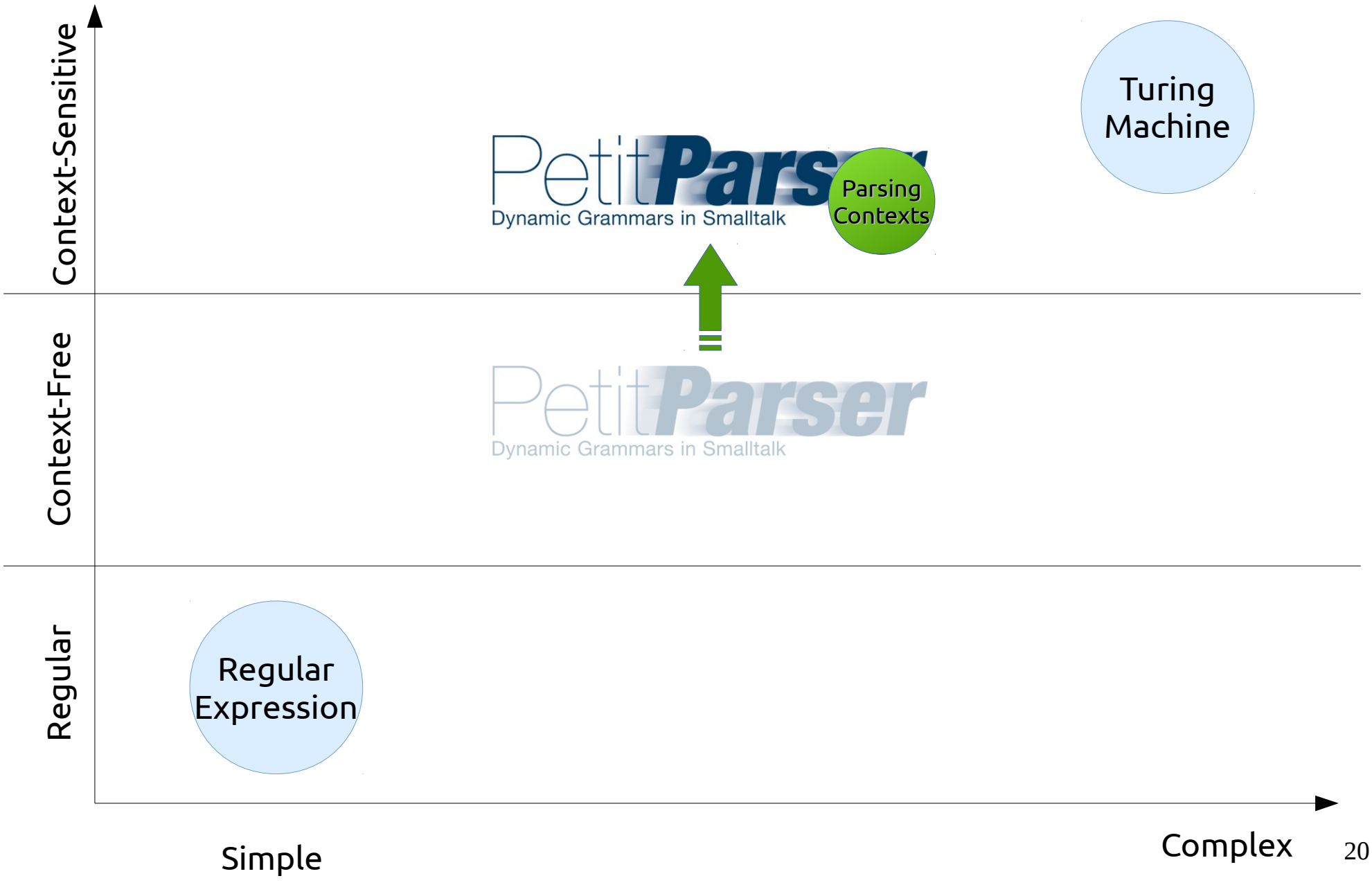
Goal


Context-Sensitive
Context-Free
Regular
Turing Machine
PetitParser
Dynamic Grammars in Smalltalk
PetitParser
Dynamic Grammars in Smalltalk
Regular Expression
Simple
Complex

Goal


Context-Sensitive
Context-Free
Regular
Turing Machine
Parsing Contexts
Regular Expression
Simple
Complex

Goal


Context-Sensitive
Context-Free
Regular
Turing Machine
PetitParser
Dynamic Grammars in Smalltalk
Parsing Contexts
PetitParser
Dynamic Grammars in Smalltalk
Regular Expression
Simple
Complex

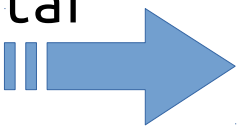
PetitParser uses **Parser Combinators**

PetitParser uses **Parser Combinators**

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

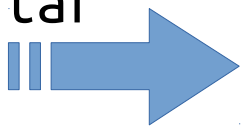
PetitParser uses **Parser Combinators**

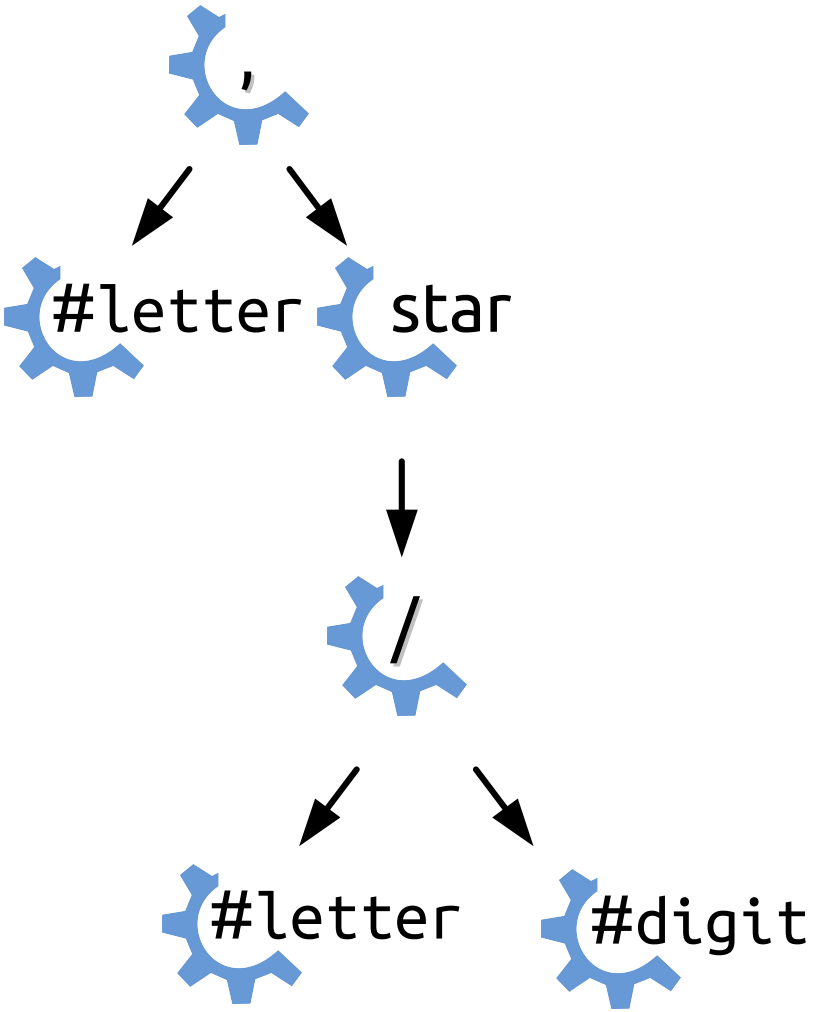
```
#letter, (#letter / #digit) star
```



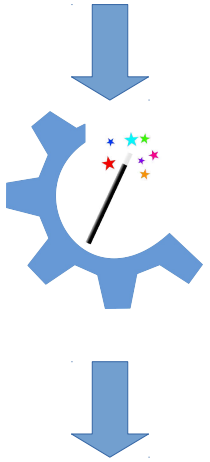
PetitParser uses **Parser Combinators**

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

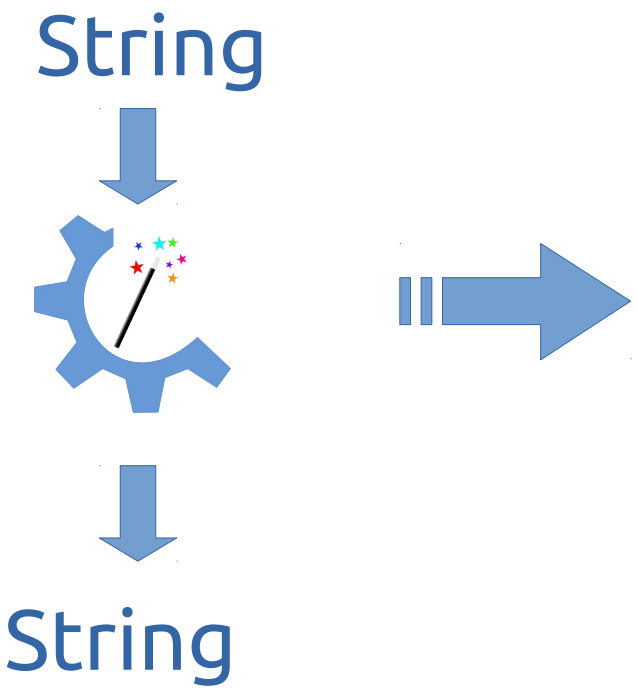


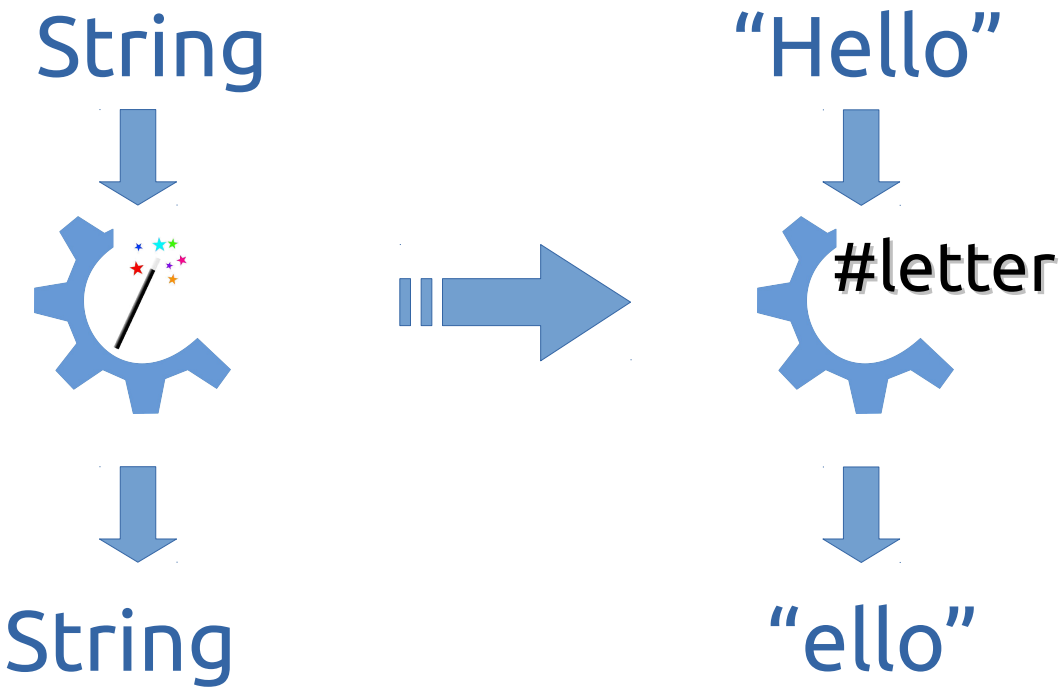


String



String


String
String


String
“Hello”
#letter
String
“ello”

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

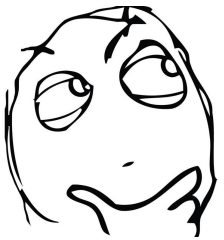
```
block  := indent, command plus, dedent.
```

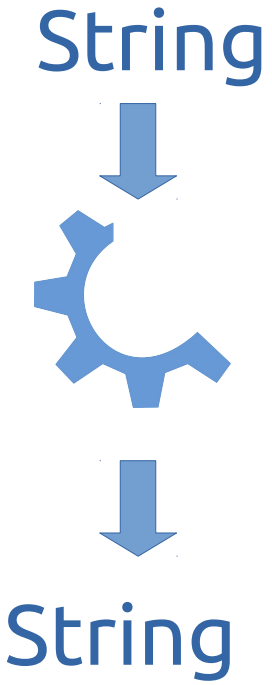
Indentation Example

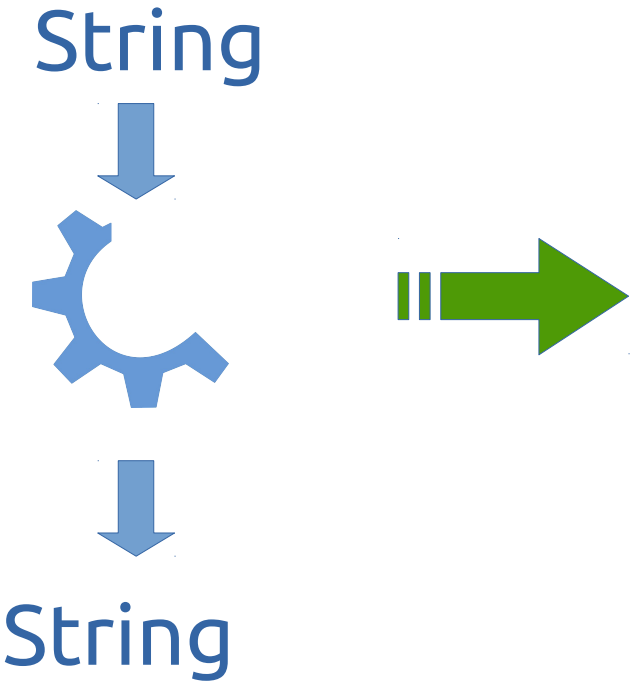
print “We are out of milk!”
order(milk)

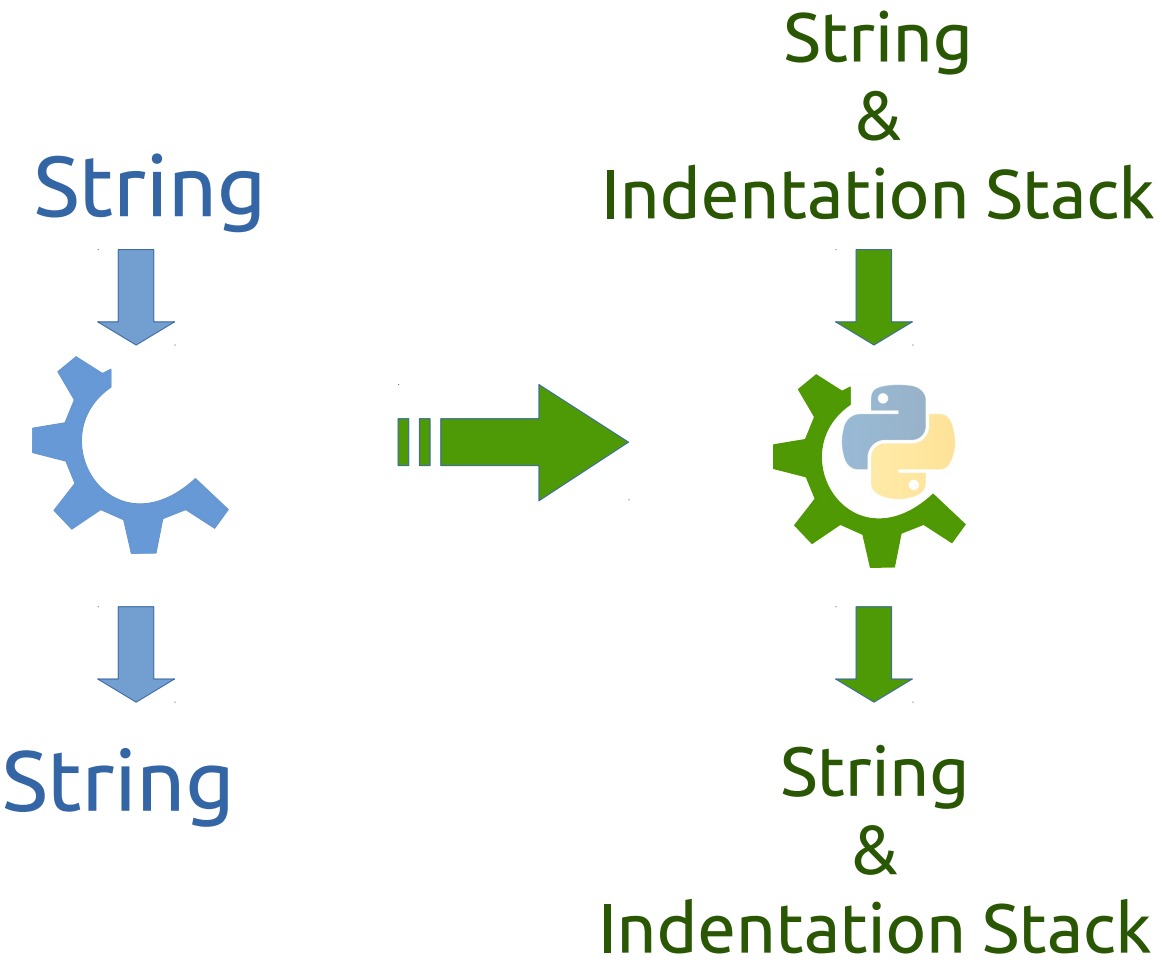
→ block of commands

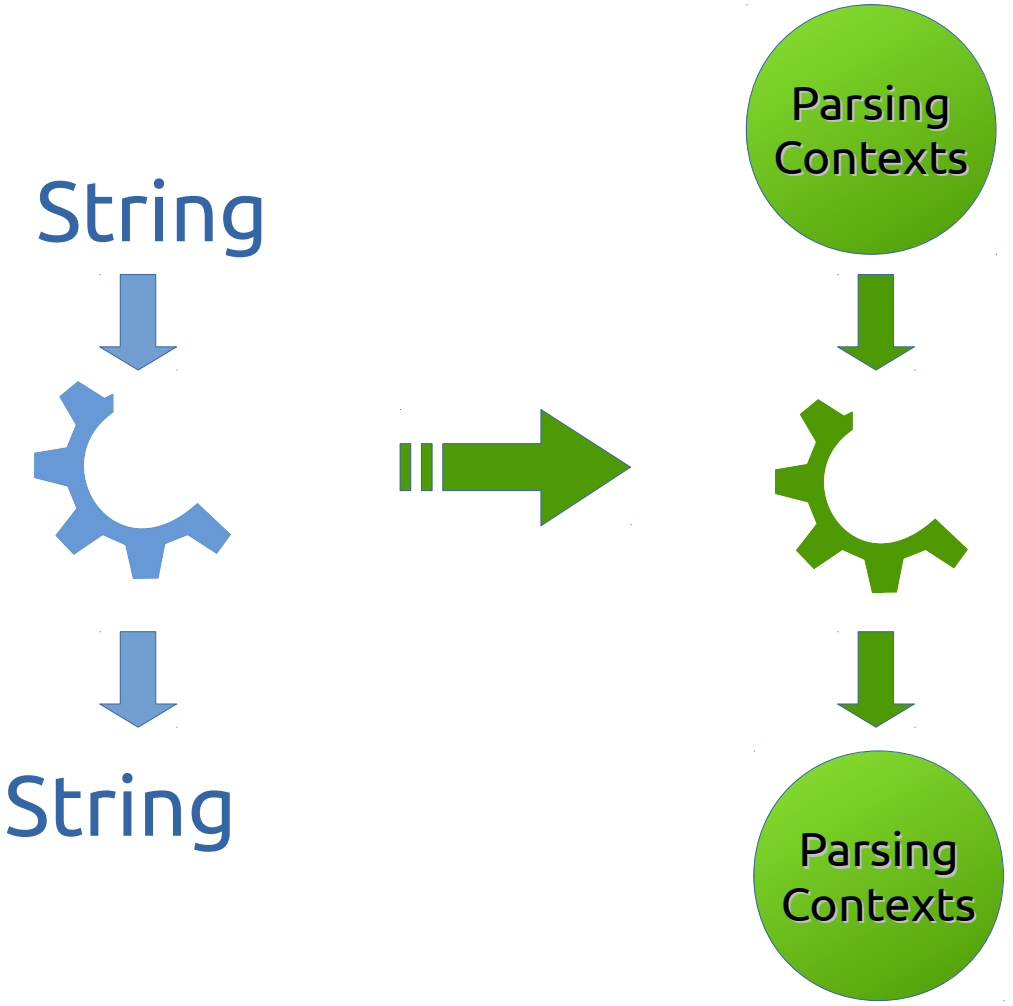
```
block  := indent, command plus, dedent.

indent := ???
dedent := ???
```

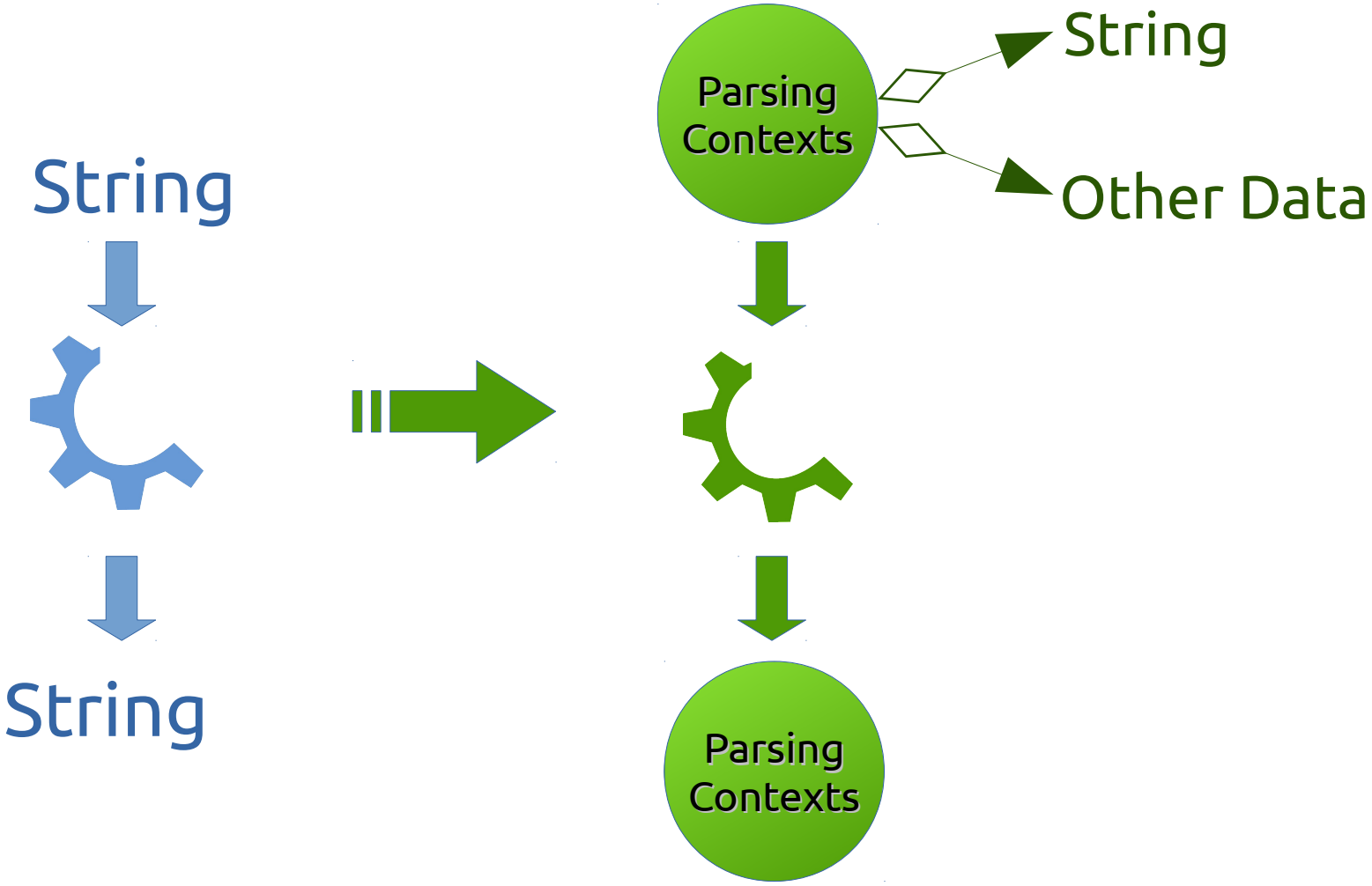



String
String


String
String


String
String
String
&
Indentation Stack
String
&
Indentation Stack

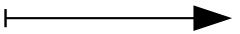

String
String
Parsing Contexts
Parsing Contexts


String
String
Parsing Contexts
String
Other Data
Parsing Contexts

Indentation Example

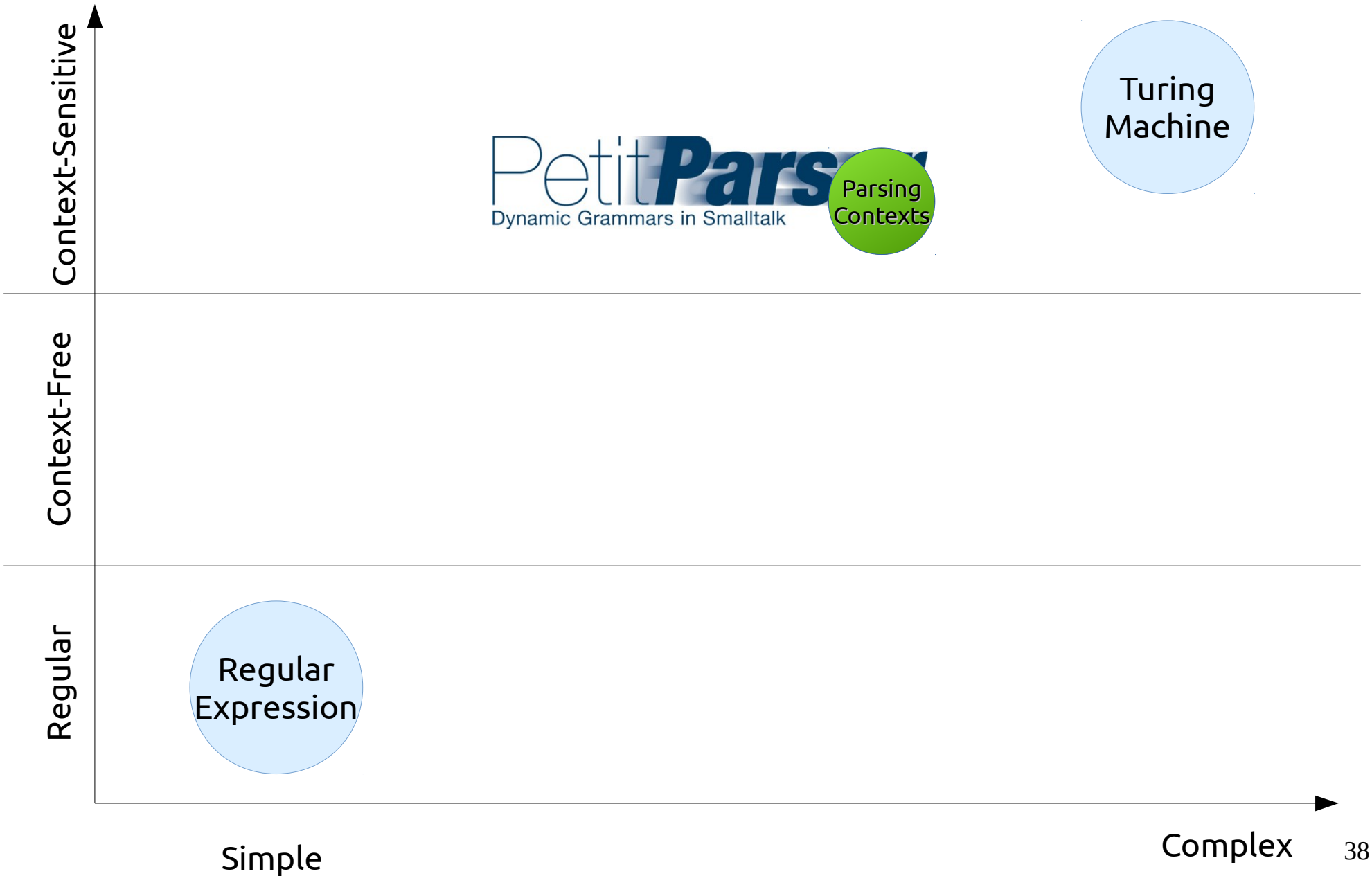
print “We are out of milk!”

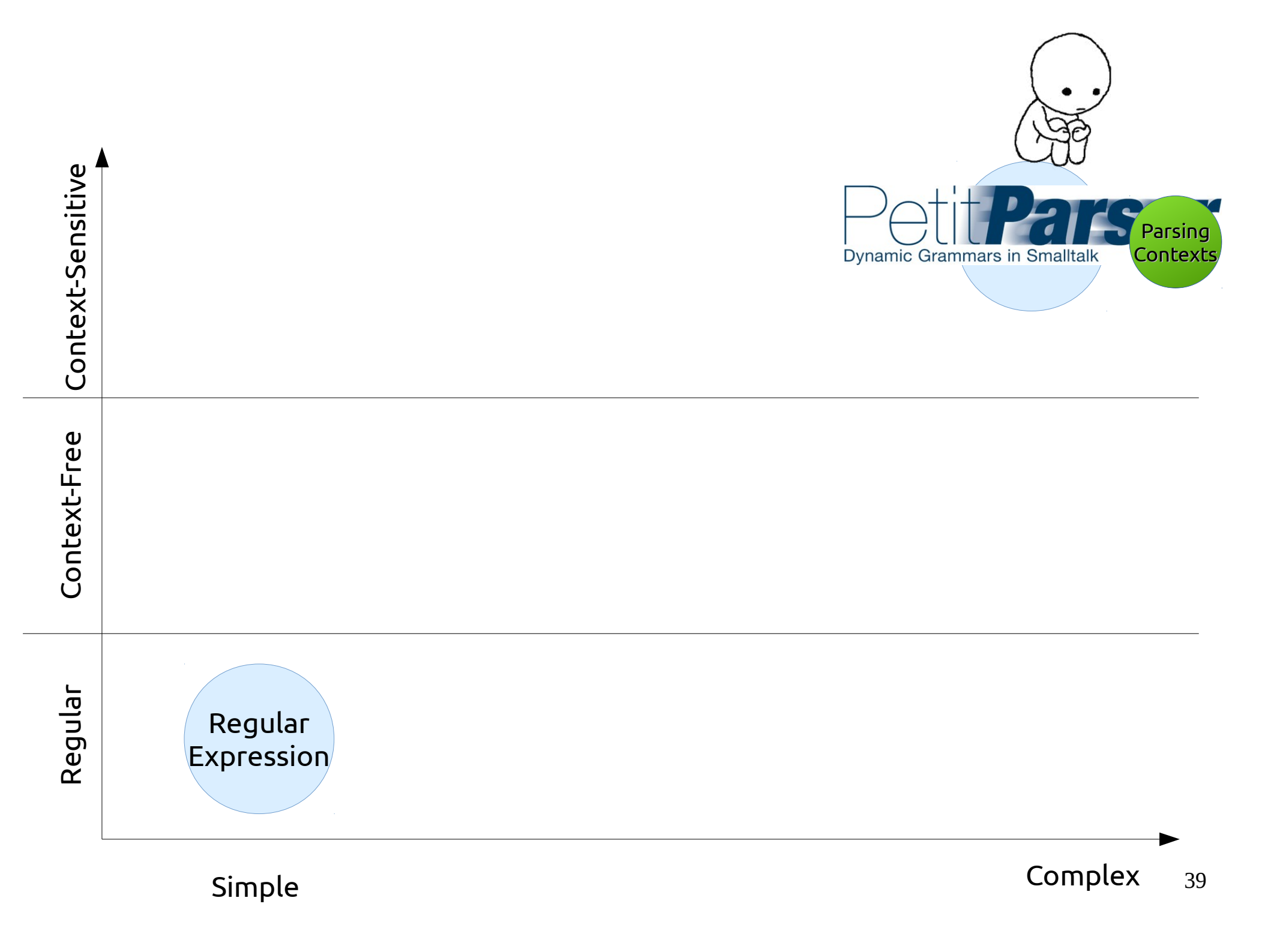
order(milk)



block of commands

```
block := [: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
PetitParser
Dynamic Grammars in Smalltalk
Parsing Contexts
Turing Machine
Regular Expression
Simple
Complex


Context-Sensitive
Context-Free
Regular
Regular Expression
Simple
Complex
PetitParser
Dynamic Grammars in Smalltalk
Parsing Contexts

Grammar
Implementors

Grammar
Implementors

Framework
Implementors

Grammar
Implementors

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

Framework
Implementors

Grammar Implementors

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

indent, command plus, dedent

Framework Implementors

Grammar Implementors

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

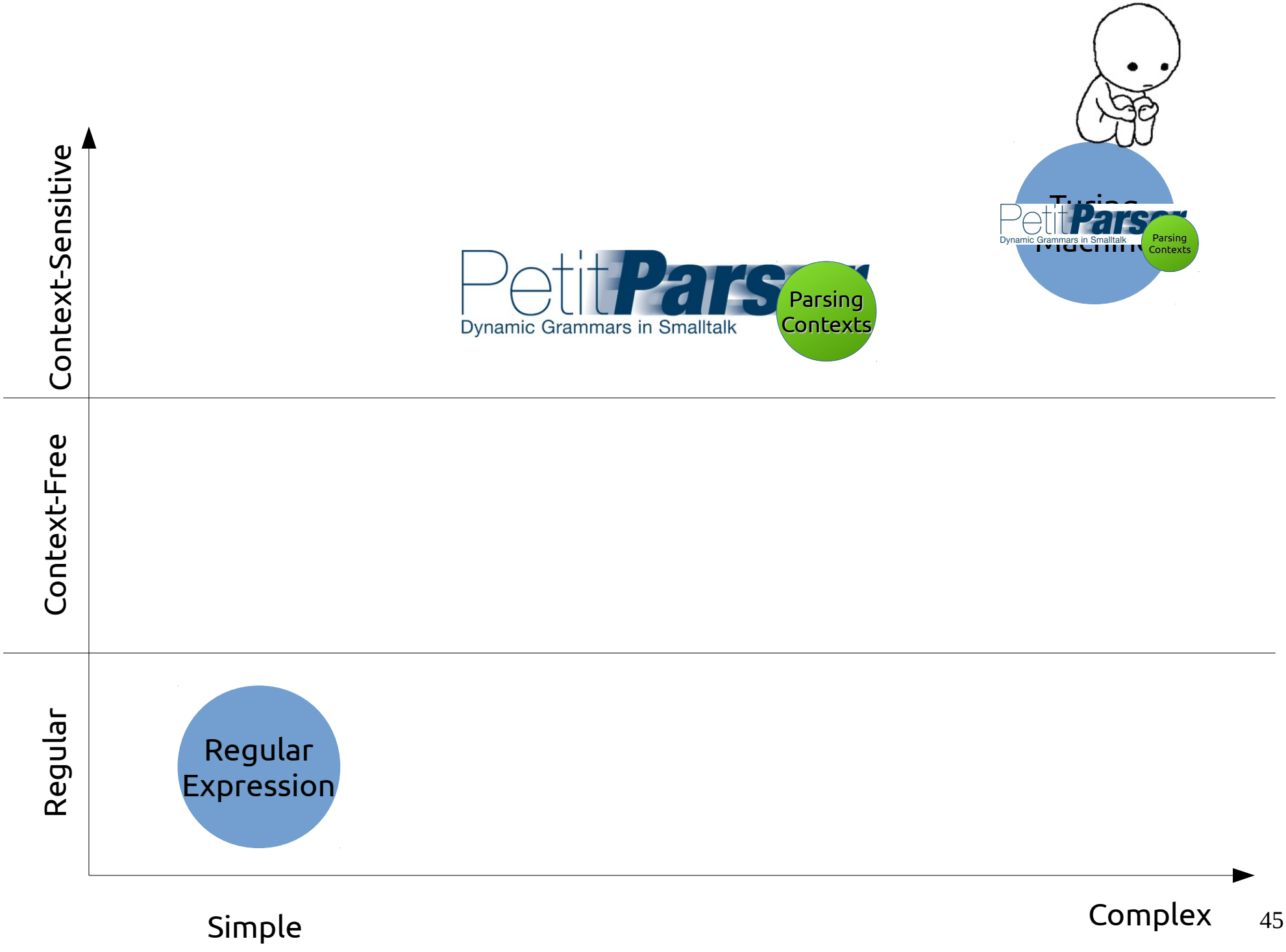
indent, command plus, dedent

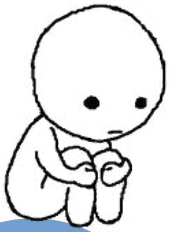
Framework Implementors

```
[: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
PetitParser
Dynamic Grammars in Smalltalk
Parsing Contexts
Regular Expression
Simple
Complex



Turing Machine

PetitParser

Dynamic Grammars in Smalltalk

Parsing Contexts

Context-Sensitive

Context-Free

Regular

Regular Expression

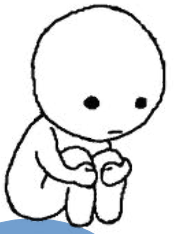
Simple

Complex

PetitParser

Dynamic Grammars in Smalltalk

Parsing Contexts



Turing Machine

Context-Sensitive

Context-Free

Regular

Regular Expression

Simple

Complex

PetitParser
Dynamic Grammars in Smalltalk

indentation

PetitParser
Dynamic Grammars in Smalltalk

indentation

PetitParser
Dynamic Grammars in Smalltalk

indentation

tolerant parsing

PetitParser
Dynamic Grammars in Smalltalk

indentation

tolerant parsing

PetitParser
Dynamic Grammars in Smalltalk


indentation


tolerant parsing


typedef

PetitParser
Dynamic Grammars in Smalltalk


indentation



tolerant parsing



typedef


PetitParser
Dynamic Grammars in Smalltalk


indentation



tolerant parsing



typedef



recursion

PetitParser
Dynamic Grammars in Smalltalk


indentation



tolerant parsing



typedef



recursion


PetitParser
Dynamic Grammars in Smalltalk

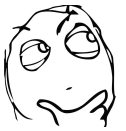
PetitParser
Dynamic Grammars in Smalltalk
Parsing Contexts

indentation

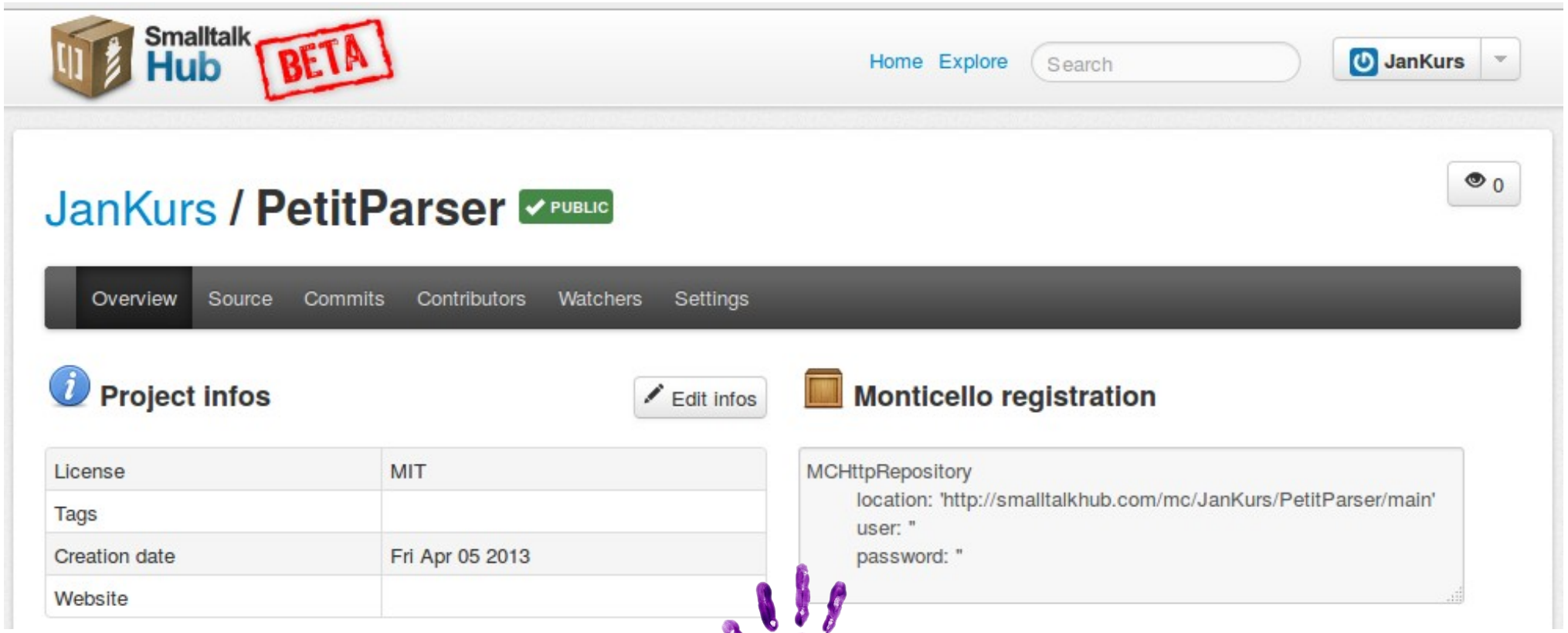
tolerant parsing

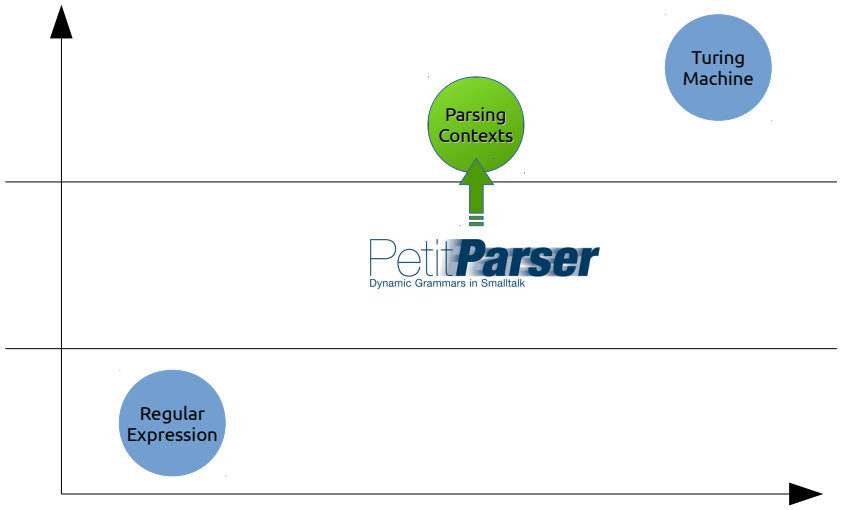
typedef

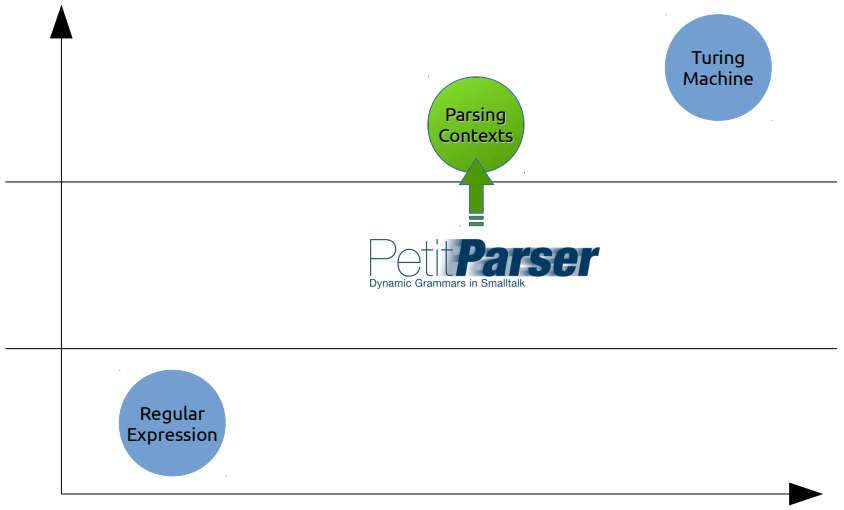
recursion

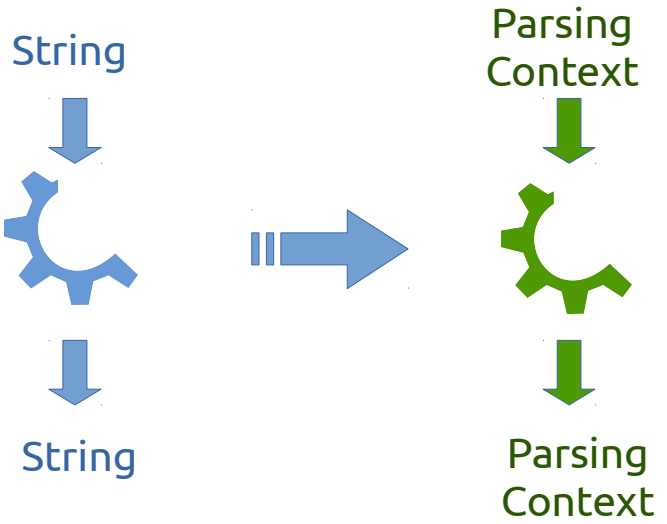


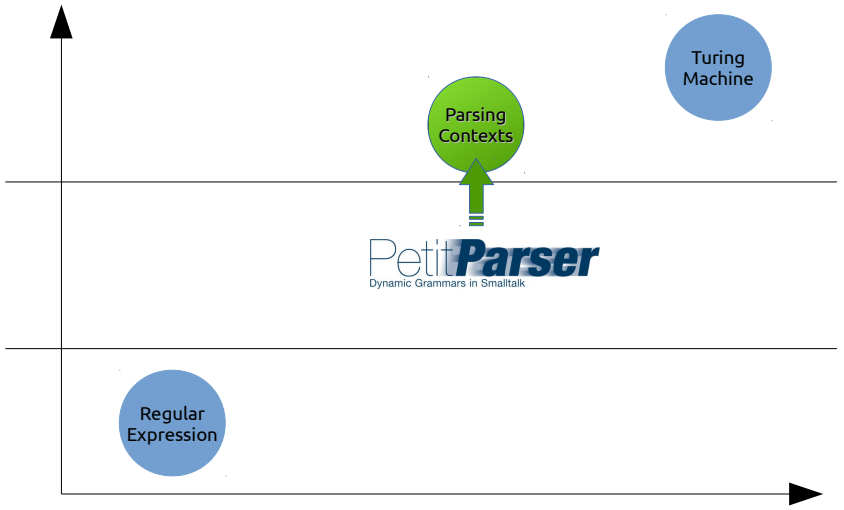
http://smalltalkhub.com/#!/~JanKurs/PetitParser

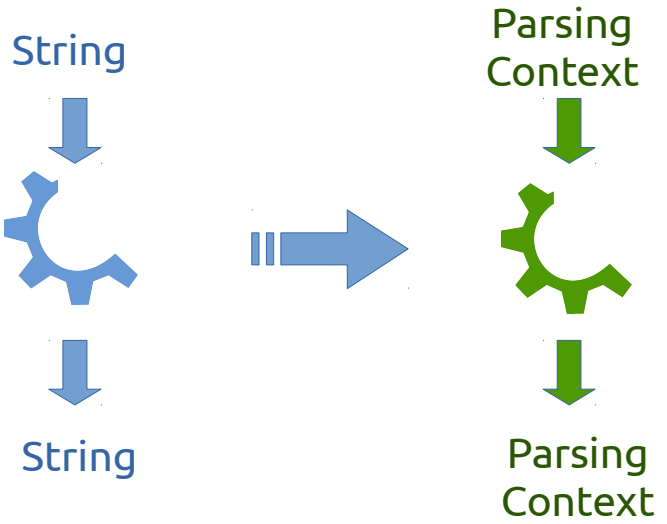



Turing
Machine
Parsing
Contexts
PetitParser
Dynamic Grammars in Smalltalk
Regular
Expression


Turing
Machine
Parsing
Contexts
PetitParser
Dynamic Grammars in Smalltalk
Regular
Expression


String
String
Parsing
Context
Parsing
Context

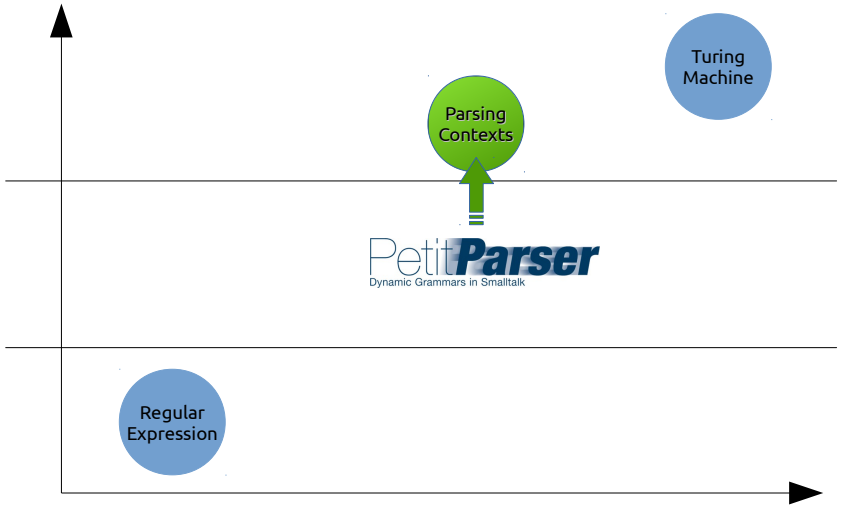


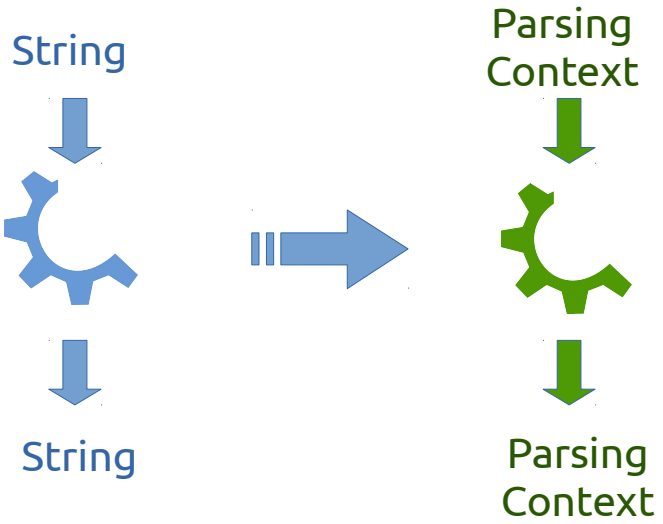


N ← (N U T)*

Grammar Implementor

Framework Implementors





N ← (N U T)*

Grammar Implementor

Framework Implementors

indentation

tolerant parsing

typedef

recursion