Learning and Refining Input Grammars for Effective Fuzzing

Rahul Gopinath

THE UNIVERSITY OF SYDNEY

Prerequisites

https://github.com/vrthra/SBST22-tutorial#readme

- Install Python 3.10
- Install Graphviz
- Install Jupyter
- Start Jupyter

Prerequisites

- Download Python 3.10

  Some variation of

```
$ sudo apt-get install python3.10
```

- Install Graphviz for your operating system.

  Some variation of

```
$ sudo apt install graphviz
```

- Make a virtual environment (recommended)

```
$ python3 -m venv sbst2022
$ cd sbt2022
$ source bin/activate
```

- Install Jupyter.

```
$ ./bin/python3 -m pip install "jupyter==1.0.0"
```

- Checkout this repository

```
git clone git@github.com:vrthra/SBST22-tutorial.git
```

- Start the Jupyter server in the repository directory

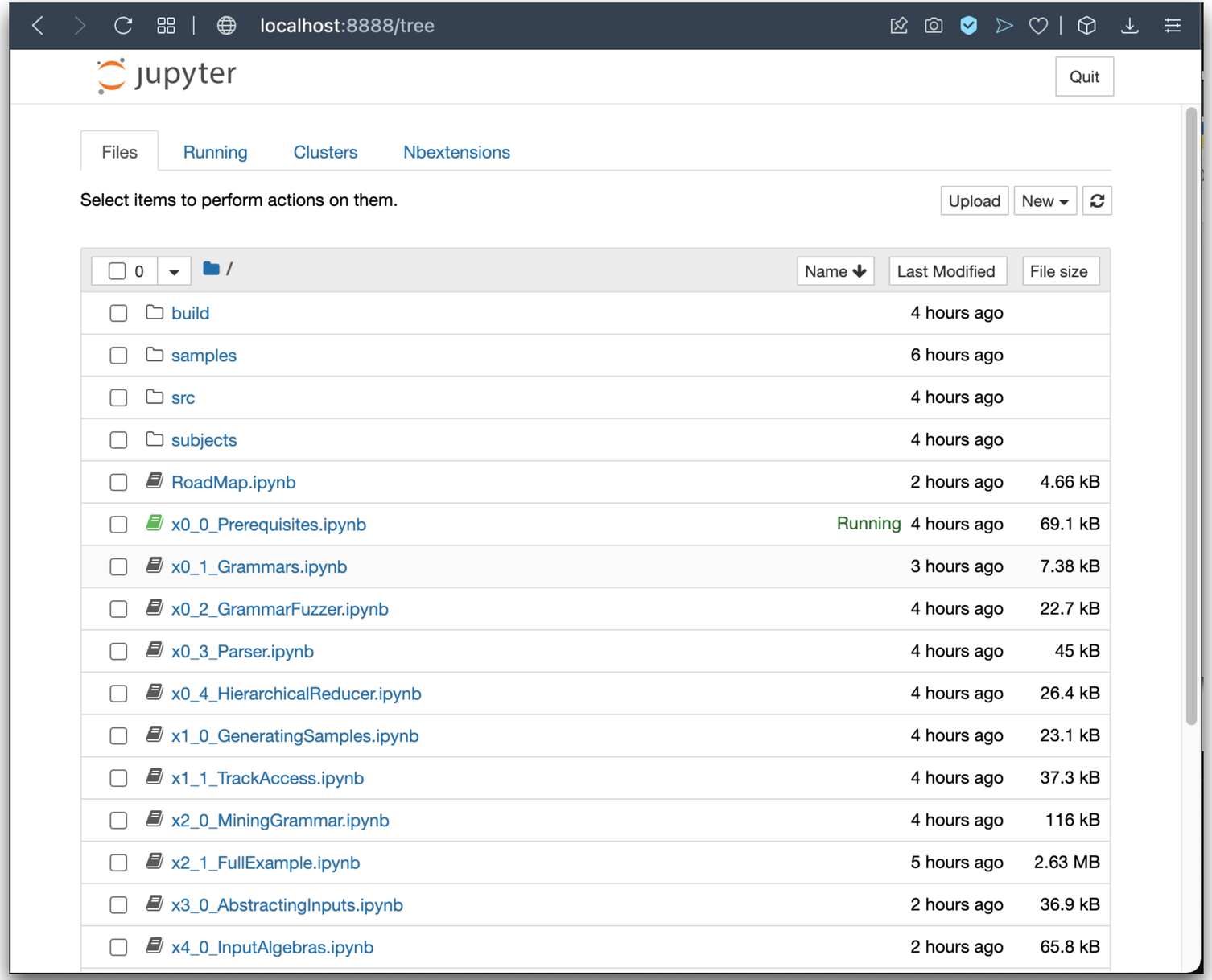
```
$ cd SBST22-tutorial
$ jupyter-notebook
```

  This opens a browser window at http://localhost:8888/tree

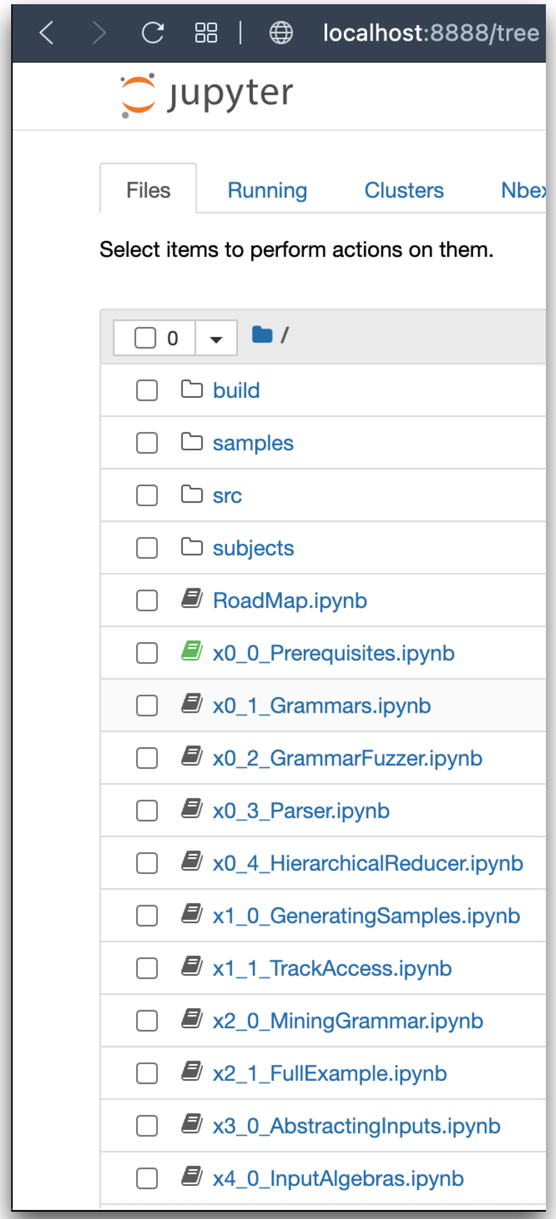
- Proceed to x0_0_Prerequisites.ipynb and execute the page completely.

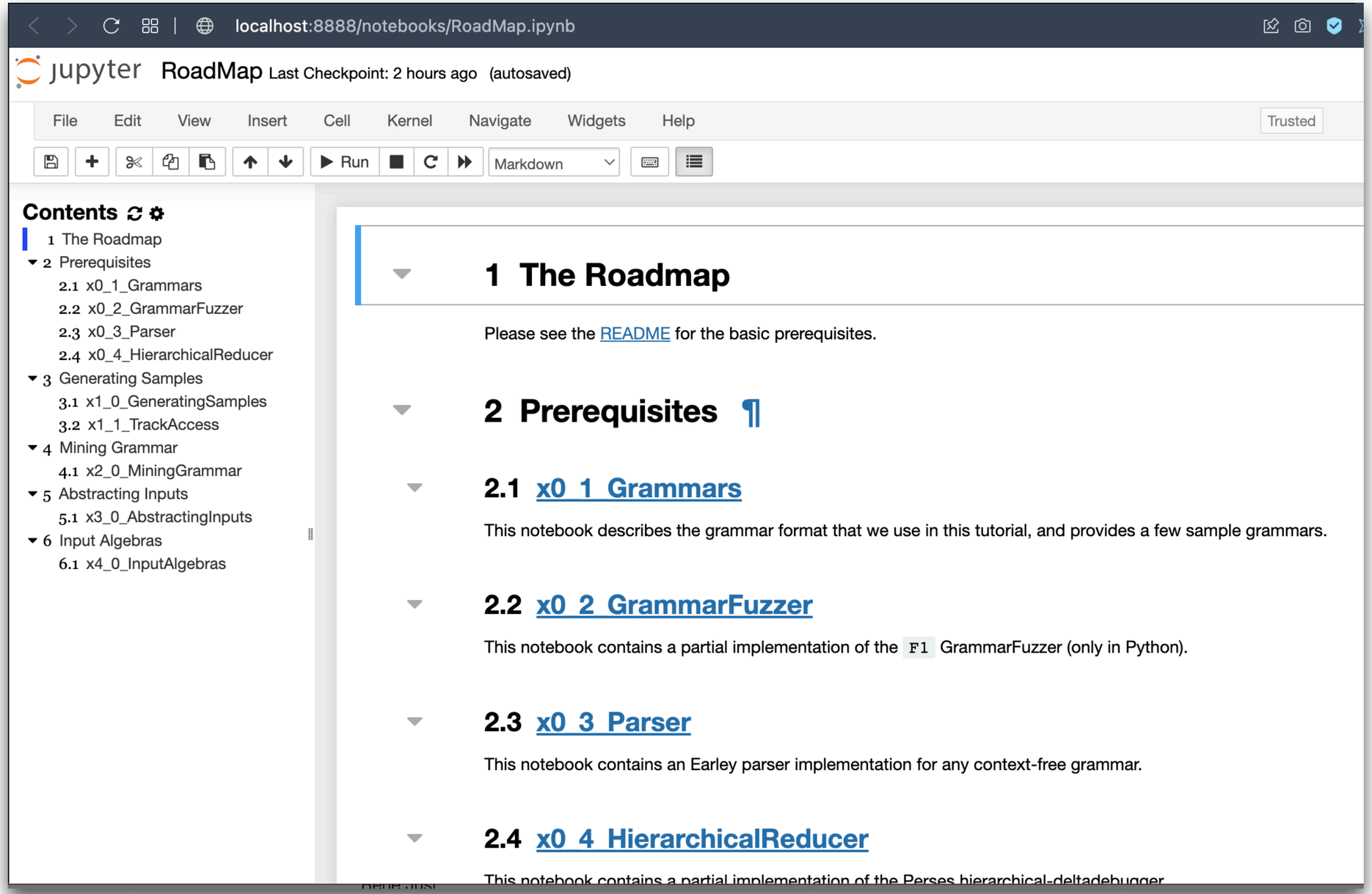
Prerequisites

- Install Python 3.10
- Install Graphviz
- Install Jupyter
- Start Jupyter

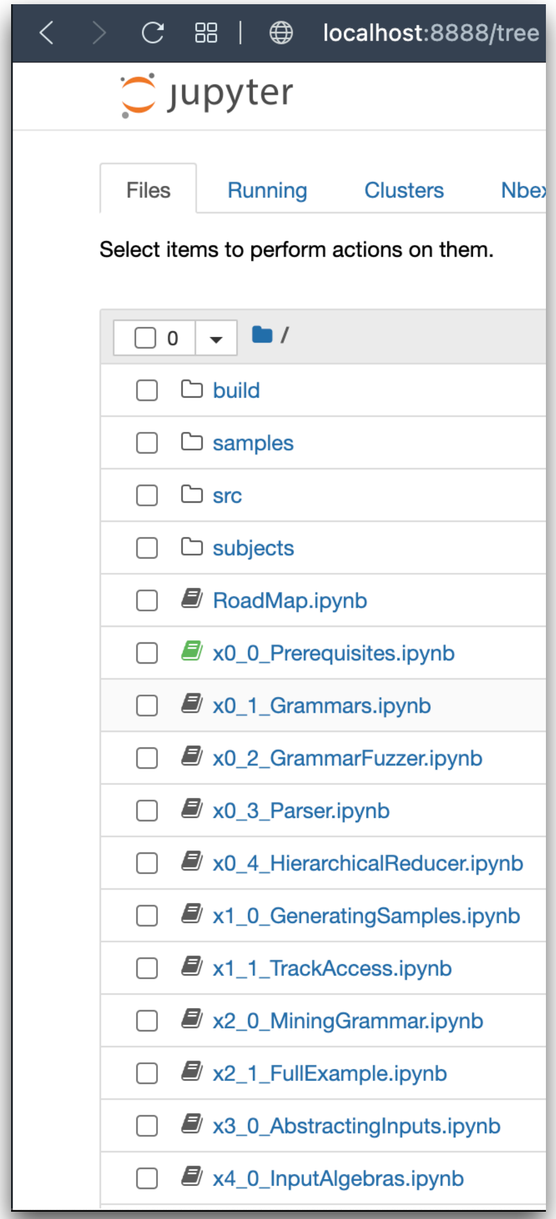


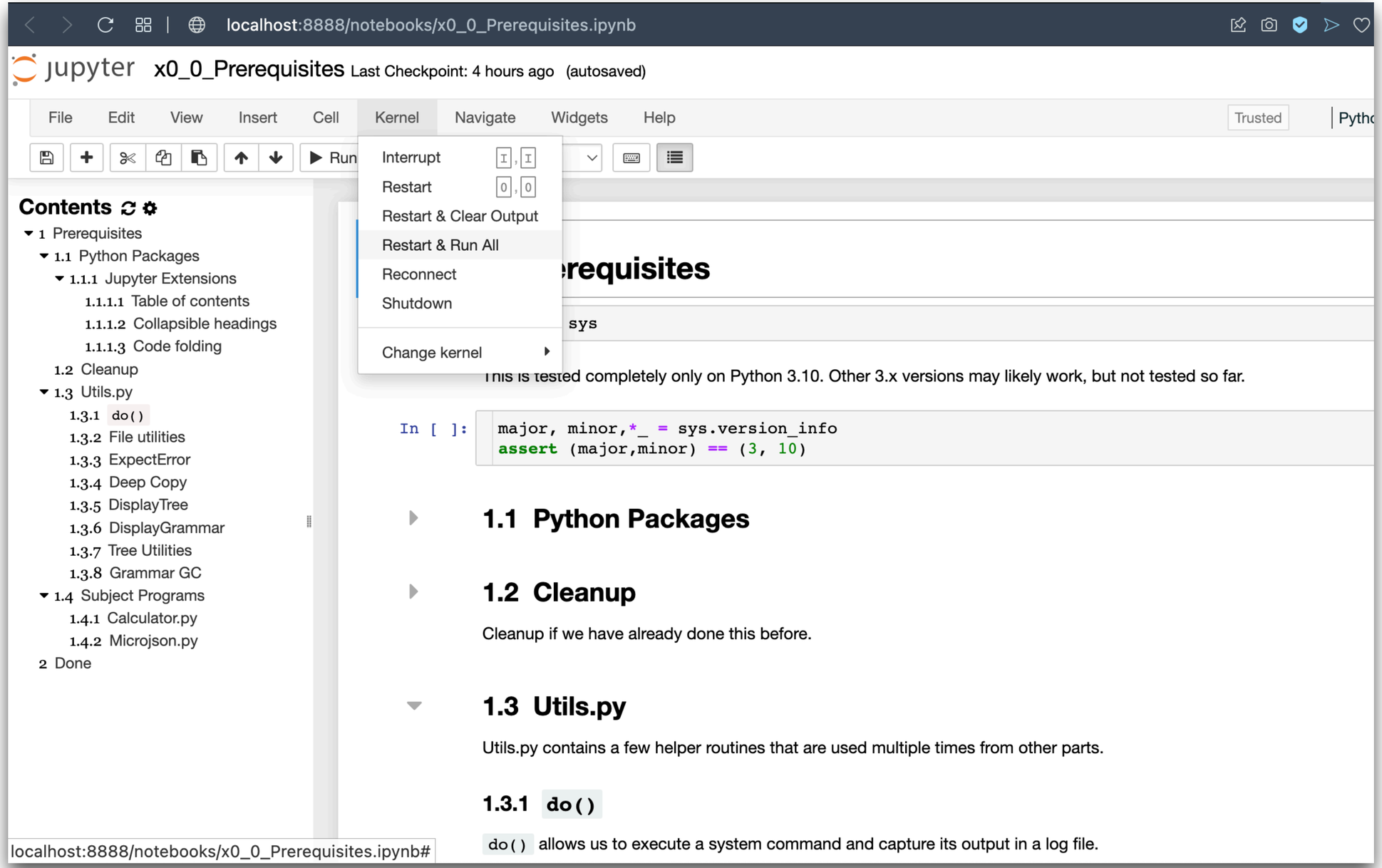
http://localhost:8888/tree





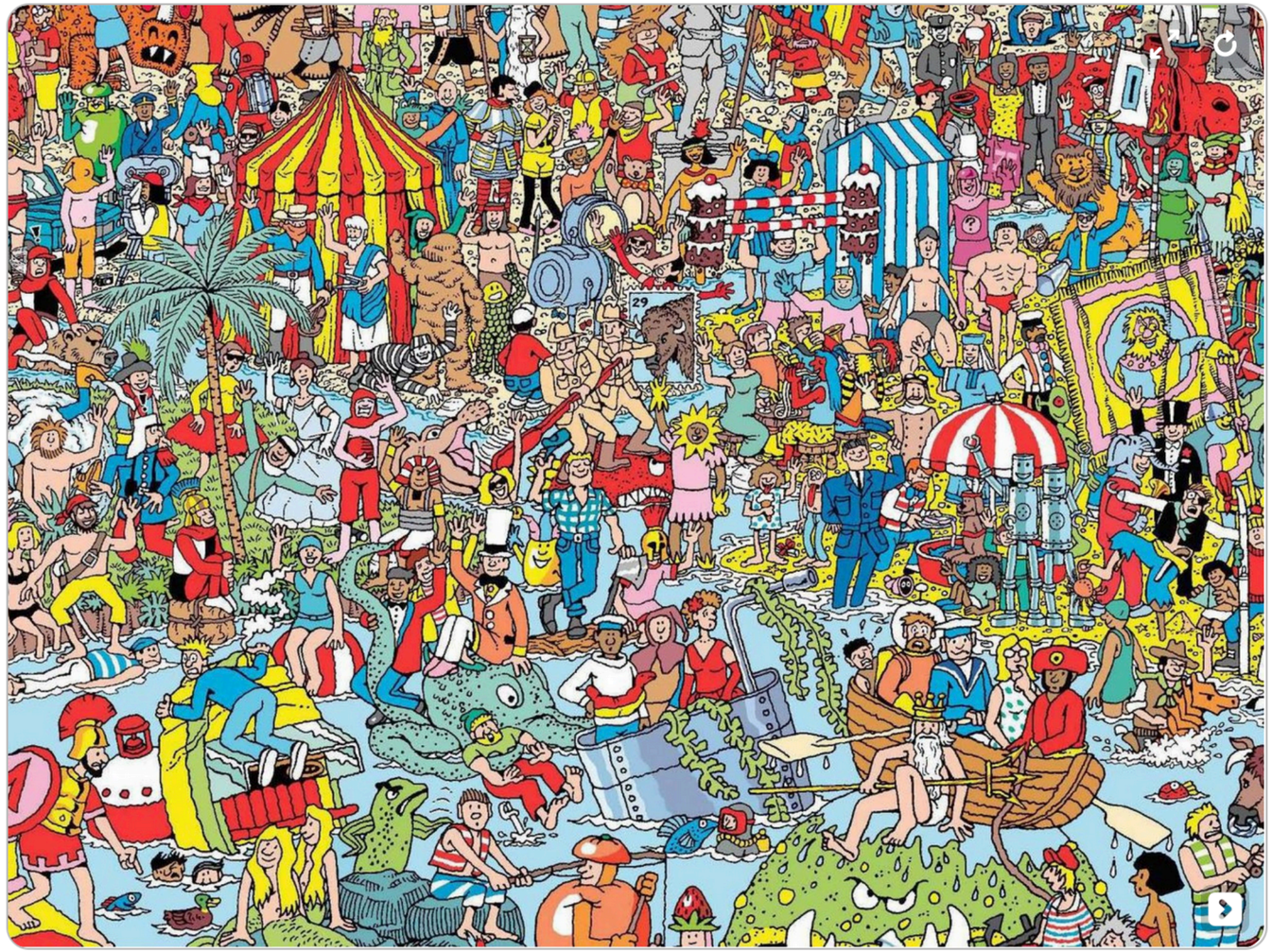
http://localhost:8888/notebooks/RoadMap.ipynb





http://localhost:8888/notebooks/x0_0_Prerequisites.ipynb

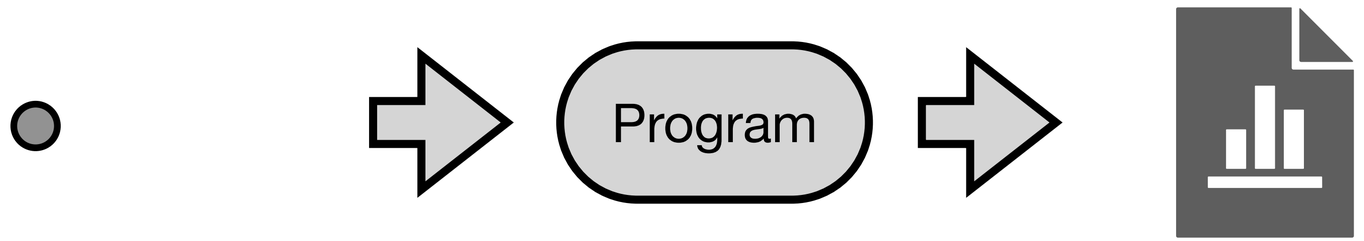
Search



Fuzzing

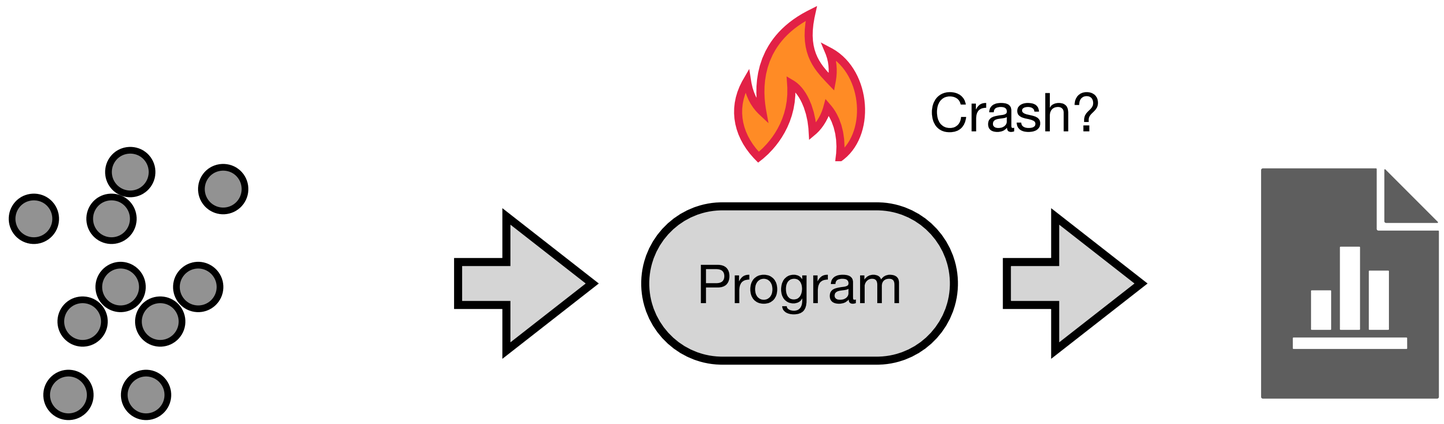
Trash deck technique: 1950s - Gerald Weinberg

Fuzzing



Trash deck technique: 1950s - Gerald Weinberg

Fuzzing



Trash deck technique: 1950s - Gerald Weinberg

Random Fuzzing

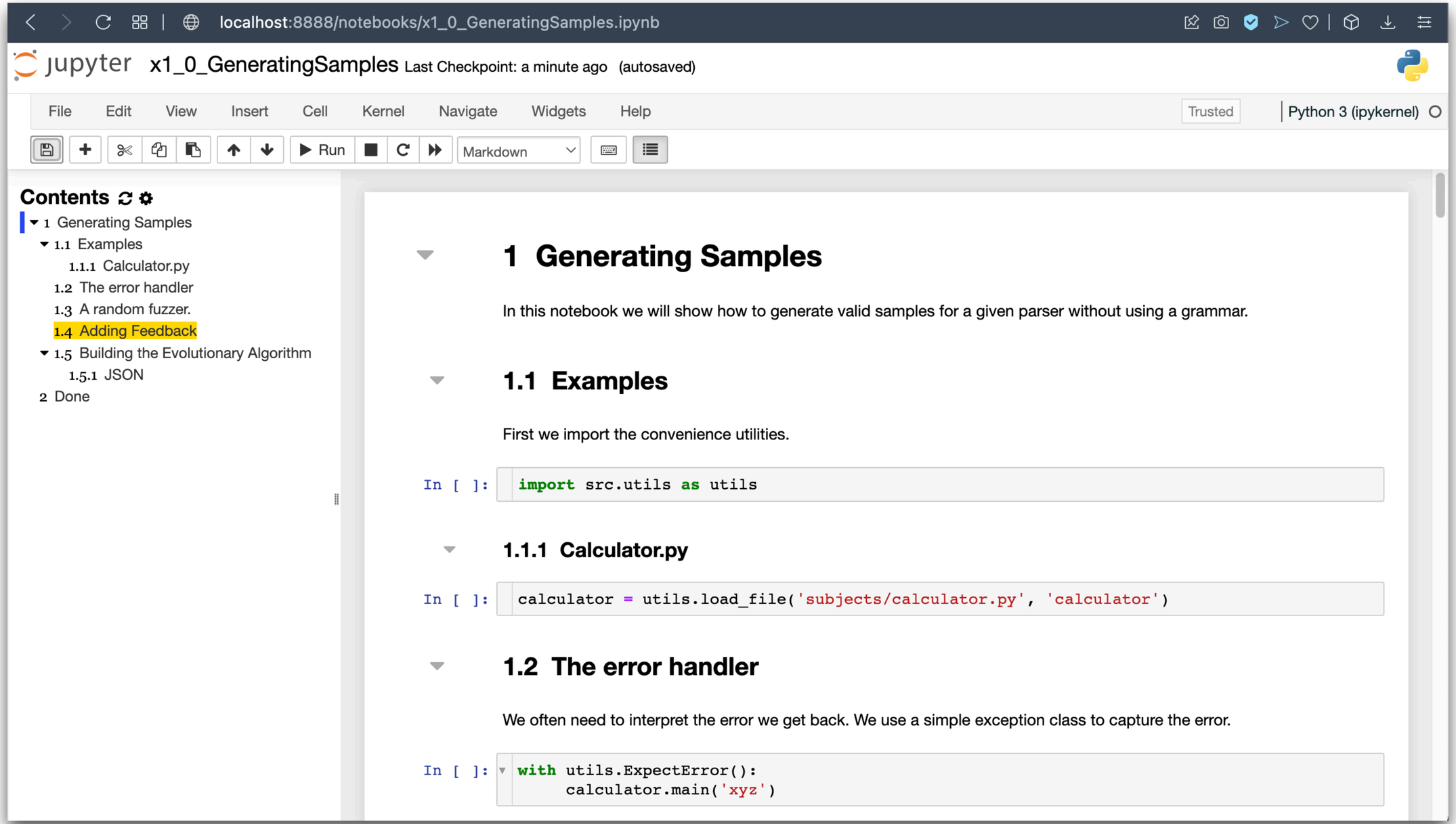
Random Fuzzing

```
$ ./fuzz

[;x1-GPZ+wcckc];,N9J+?#6^6\e?]9lu
2_%'4GX"0VUB[E/r ~fApu6b8<{%siq8Z
h.6{V,hr?;{Ti.r3PIxMMMv6{xS^+'Hq!
AxB"YXRS@!Kd6;wtAMefFWM(`|J_<1~o}
z3K(CCzRH JIIvHz>_*.\>JrlU32~eGP?
lR=bF3+;y$3lodQ<B89!5"W2fK*vE7v{'
)KC-i,c{<[~m!]o;{.'}Gj\(X}EtYetrp
bY@aGZ1{P!AZU7x#4(Rtn!q4nCwqol^y6
}0|Ko=*JK~;zMKV=9Nai:wxu{J&UV#HaU
)*BiC<),`+t*gka<W=Z.%T5WGHZpI30D<
Pq>&]BS6R&j?#tP7iaV}-}`\?[_[Z^LBM
PG-FKj'\xwuZ1=Q`^`5,$N$Q@[!CuRzJ2
D|vBy!^zkhdf3C5PAkR?V((-%><hn|3='
i2Qx]D$qs4O`1@fevnG'2\11Vf3piU37@
5:dfd45*(7^%5ap\zIyl"'f,$ee,J4Gw:
cgNKLie3nx9(`efSlg6#[K"@WjhZ}r[Sc
un&sBCS,T[/3]KAeEnQ7lU)3Pn,0)G/6N
-wyzj/MTd#A;r*(ds./df3r8Odaf?/<#r
```

Feedback Driven Fuzzing

```
$ ./fuzz -int | program

6341115697428101937274240695O9
7413559250614994511624647195O6
6159573319248265555905374076O5
1814000798034468742520463747l6
7409737702553482794256013331l4
1527240579320738285690412161l1
0998594464965099190248102712l2
6229749886714219380124646301l8
7353551345993272409202596752l3
5745286130570842313707419209O2
7946778421646549903535755804l3
7772823058553523781190380964l6
6998713066550849533770398623l7
9249575543898783529345476642l0
0824315560938372885972626755l8
6308519190618298850488347388l2
6770224294149809170539399707l5
7220069879160886501686654717l1 yes
```

```
def is_prime(n: int) -> bool:
    """Primality test using 6k+-1 optimization."""
    if n <= 3:
        return n > 1
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i ** 2 <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

def main():
    num = stdin.read()
    print(num, is_prime(num))
```



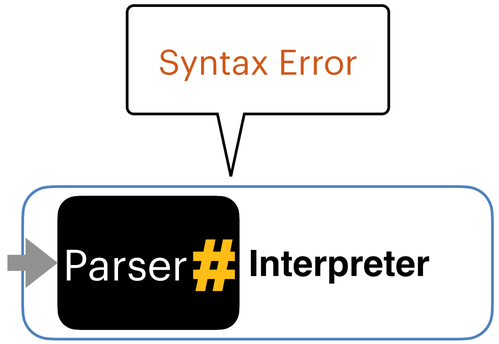
http://localhost:8888/notebooks/x1_0_GeneratingSamples.ipynb

Fuzzing Parsers

```
$ ./fuzz

[;x1-GPZ+wcckc];,N9J+?#6^6\e?]9lu
2_%'4GX"0VUB[E/r ~fApu6b8<{%siq8Z
h.6{V,hr?;{Ti.r3PIxMMMv6{xS^+'Hq!
AxB"YXRS@!Kd6;wtAMefFWM(`|J_<1~o}
z3K(CCzRH JIIvHz>_*.\>JrlU32~eGP?
lR=bF3+;y$3lodQ<B89!5"W2fK*vE7v{'
)KC-i,c{<[~m!]o;{.'}Gj\(X}EtYetrp
bY@aGZ1{P!AZU7x#4(Rtn!q4nCwqol^y6
}0|Ko=*JK~;zMKV=9Nai:wxu{J&UV#HaU
)*BiC<),`+t*gka<W=Z.%T5WGHZpI30D<
Pq>&]BS6R&j?#tP7iaV}-}`\?[_[Z^LBM
PG-FKj'\xwuZ1=Q`^`5,$N$Q@[!CuRzJ2
D|vBy!^zkhdf3C5PAkR?V((-%><hn|3='
i2Qx]D$qs4O`1@fevnG'2\11Vf3piU37@
5:dfd45*(7^%5ap\zIyl"'f,$ee,J4Gw:
cgNKLie3nx9(`efSlg6#[K"@WjhZ}r[Sc
un&sBCS,T[/3]KAeEnQ7lU)3Pn,0)G/6N
-wyzj/MTd#A;r*(ds./df3r8Odaf?/<#r
```

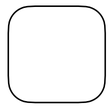
Fuzzing Parsers

```
$ ./fuzz

[;x1-GPZ+wcckc];,N9J+?#6^6\e?]9lu
2_%'4GX"0VUB[E/r ~fApu6b8<{%siq8Z
h.6{V,hr?;{Ti.r3PIxMMMv6{xS^+'Hq!
AxB"YXRS@!Kd6;wtAMefFWM(`|J_<1~o}
z3K(CCzRH JIIvHz>_*.\>JrlU32~eGP?
lR=bF3+;y$3lodQ<B89!5"W2fK*vE7v{'
)KC-i,c{<[~m!]o;{.'}Gj\(X}EtYetrp
bY@aGZ1{P!AZU7x#4(Rtn!q4nCwqol^y6
}0|Ko=*JK~;zMKV=9Nai:wxu{J&UV#HaU
)*BiC<),`+t*gka<W=Z.%T5WGHZpI30D<
Pq>&]BS6R&j?#tP7iaV}-}`\?[_[Z^LBM
PG-FKj'\xwuZ1=Q`^`5,$N$Q@[!CuRzJ2
D|vBy!^zkhdf3C5PAkR?V((-%><hn|3='
i2Qx]D$qs4O`1@fevnG'2\11Vf3piU37@
5:dfd45*(7^%5ap\zIyl"'f,$ee,J4Gw:
cgNKLie3nx9(`efSlg6#[K"@WjhZ}r[Sc
un&sBCS,T[/3]KAeEnQ7lU)3Pn,0)G/6N
-wyzj/MTd#A;r*(ds./df3r8Odaf?/<#r
```

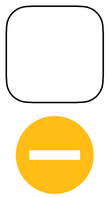


Sepcification Free Generators

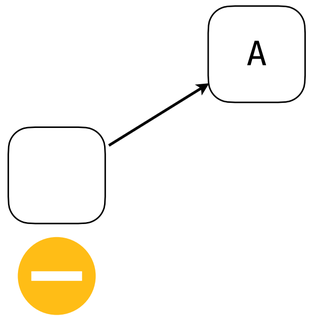
Sepcification Free Generators



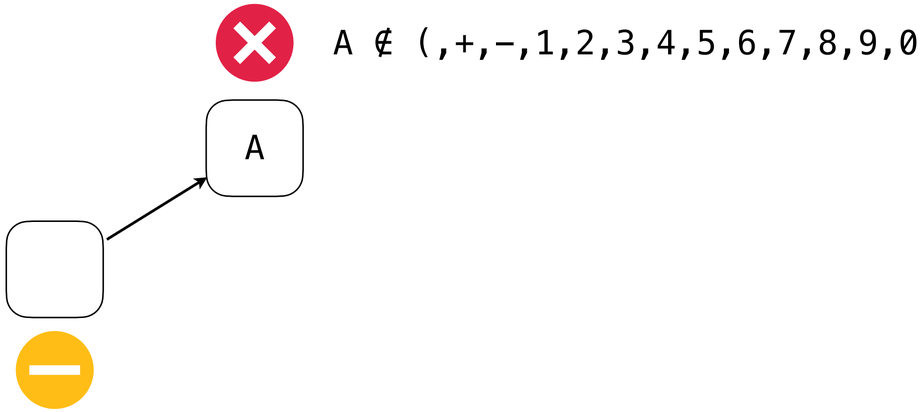
Sepcification Free Generators



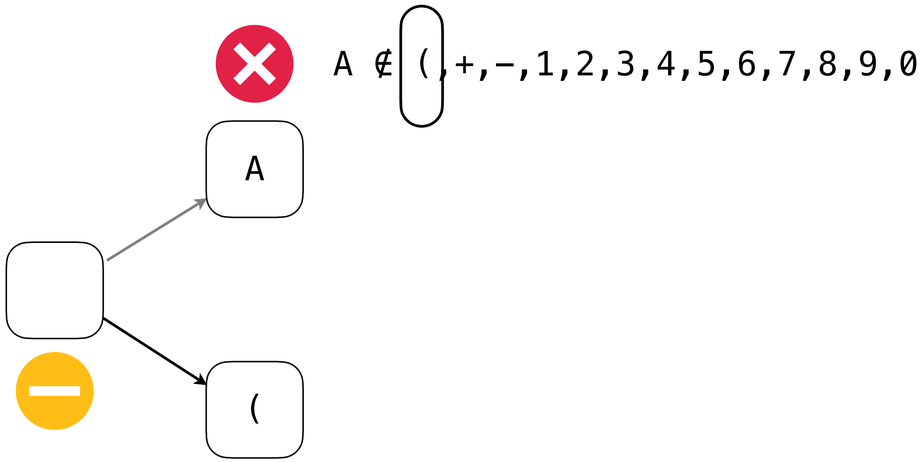
Sepcification Free Generators



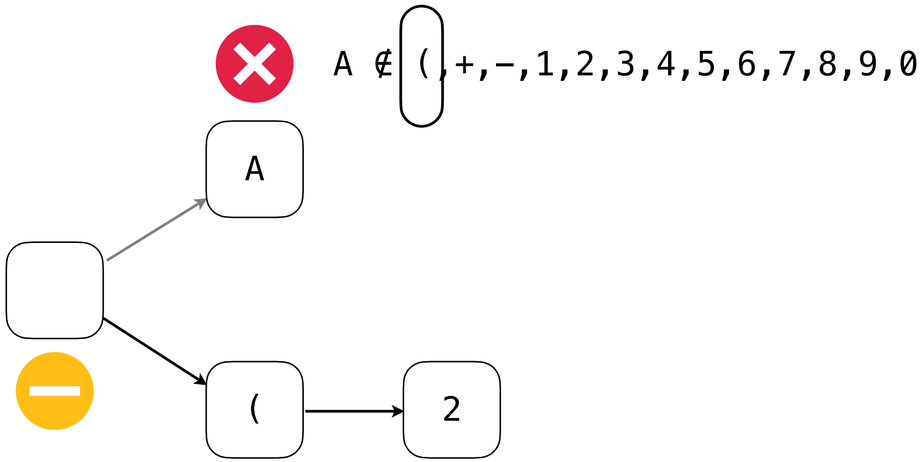
Sepcification Free Generators



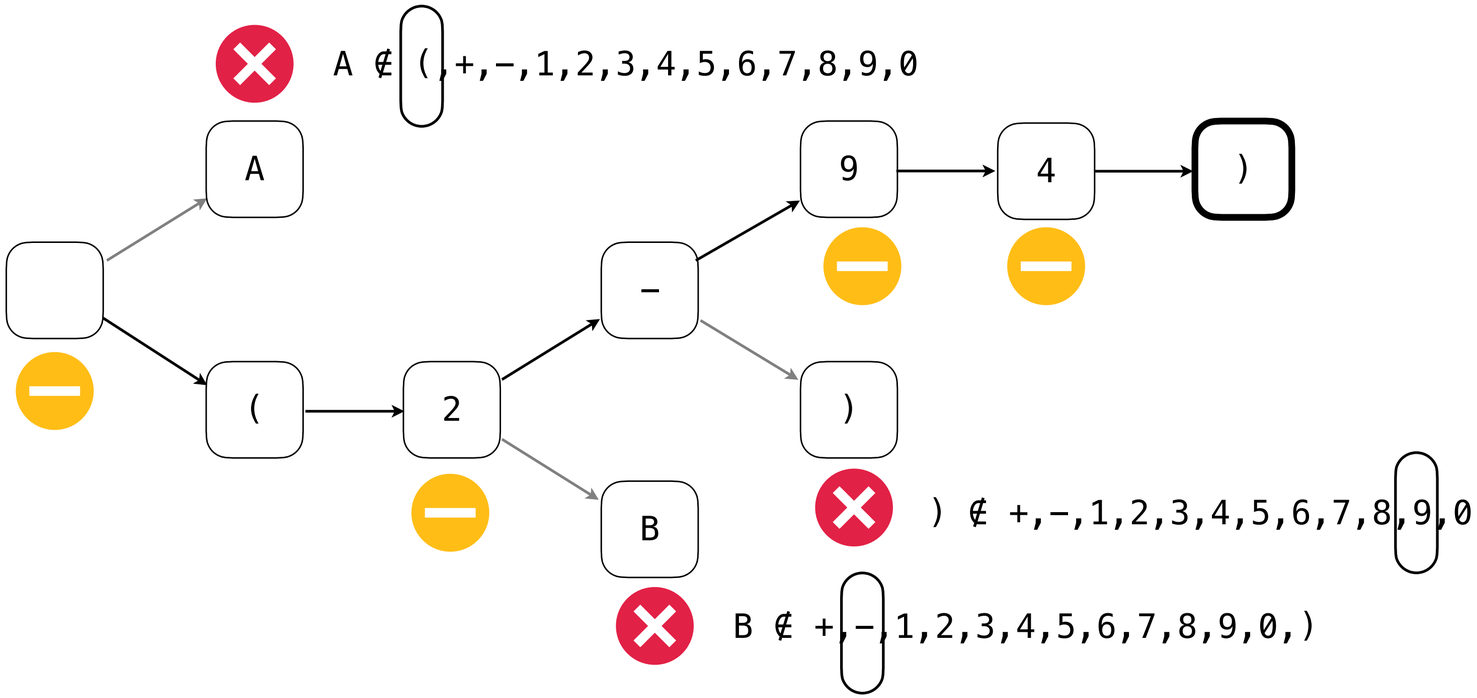
Sepcification Free Generators



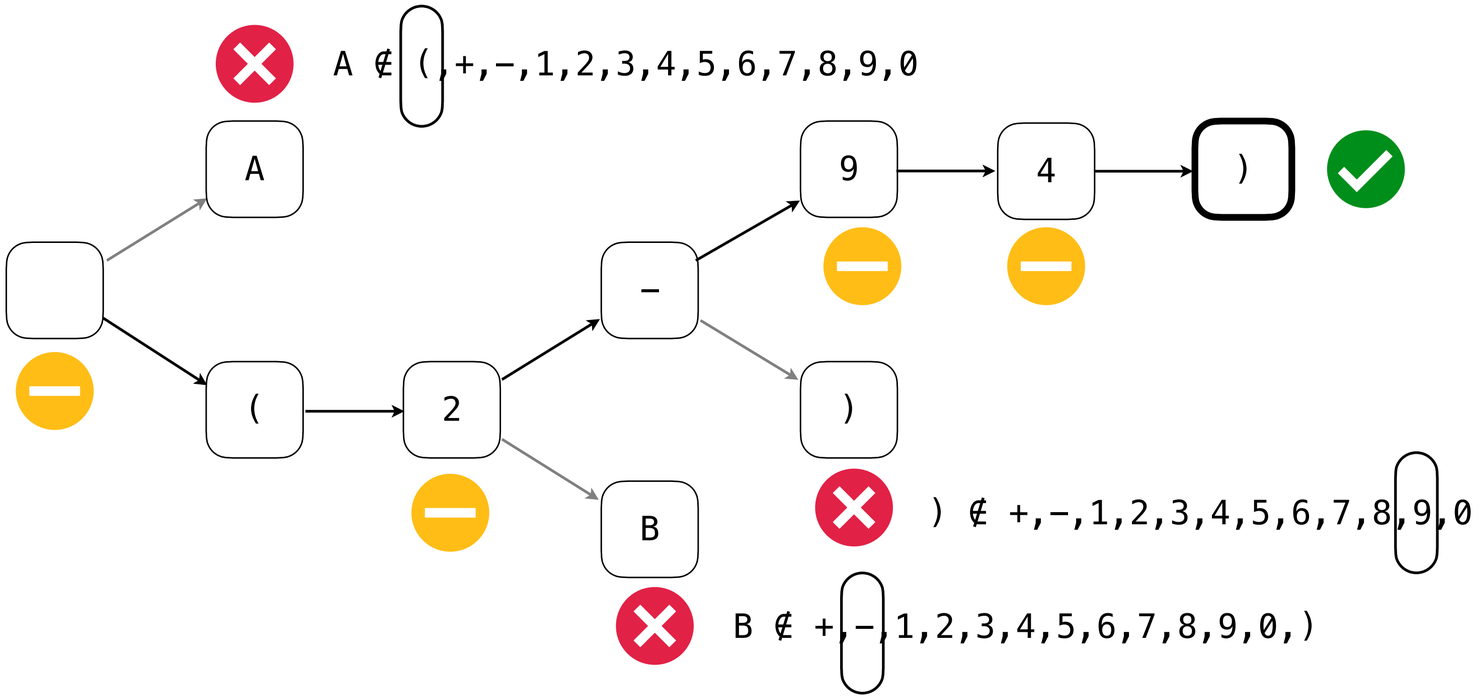
Sepcification Free Generators



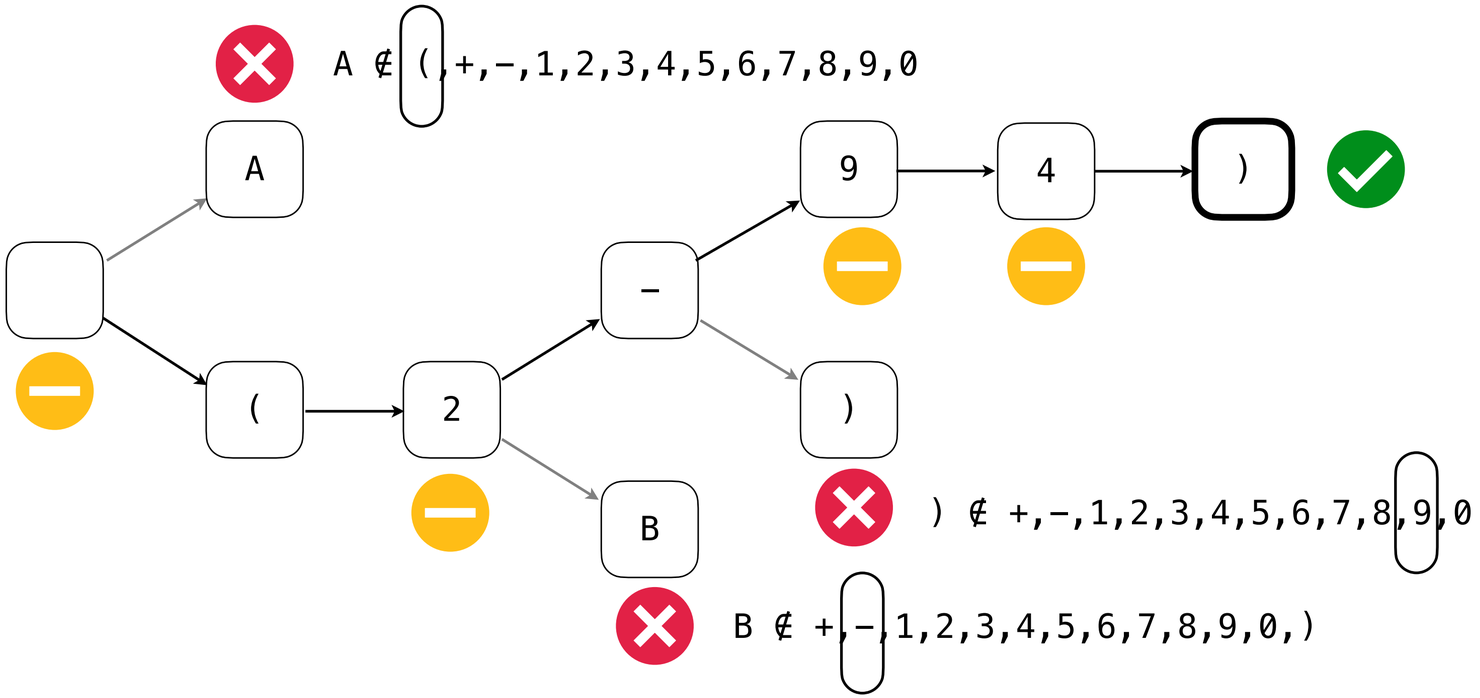
Sepcification Free Generators



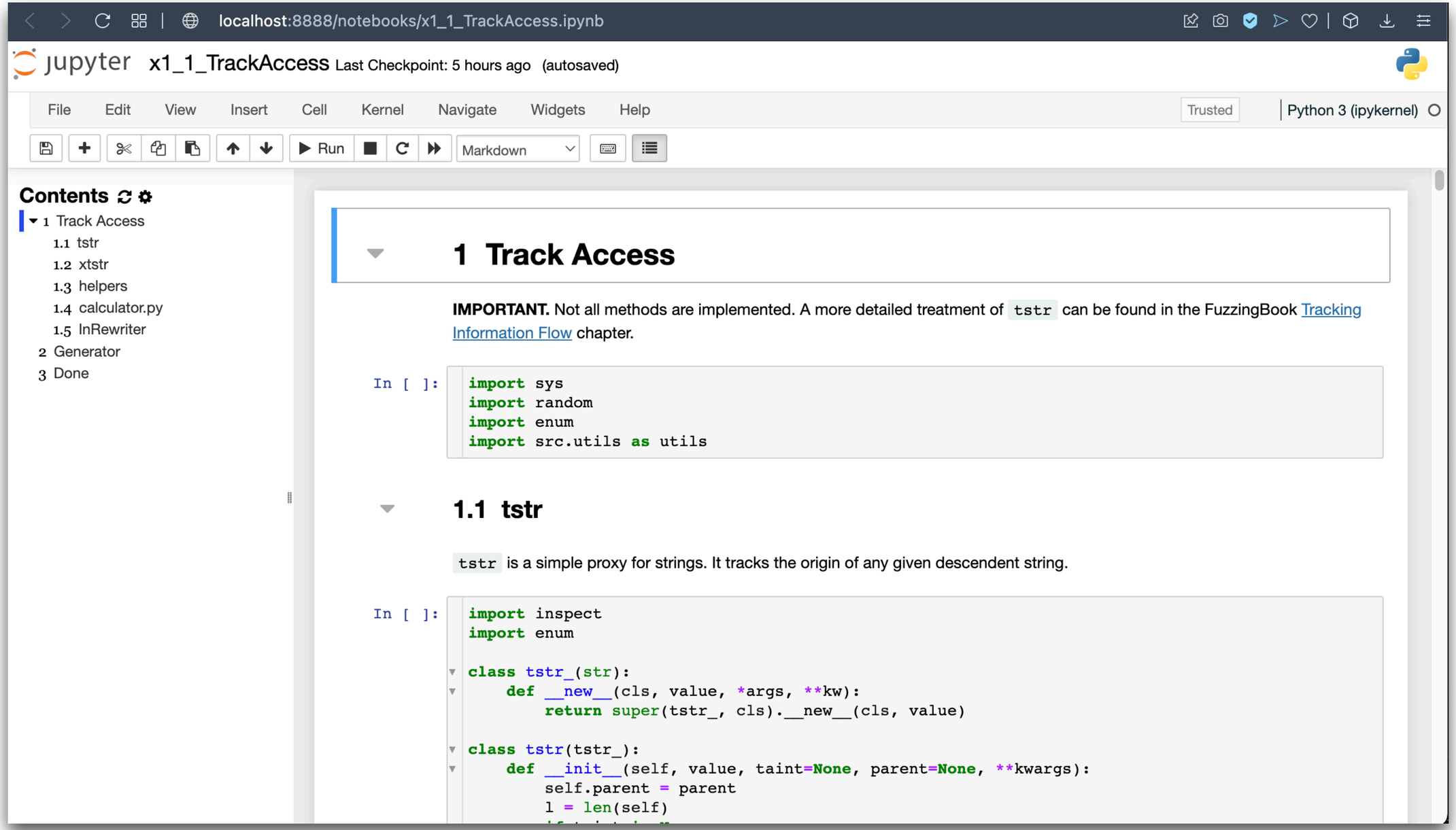
Sepcification Free Generators



Sepcification Free Generators



(2–94)



http://localhost:8888/notebooks/x1_1_TrackingAccess.ipynb

Limitation: Lack of control

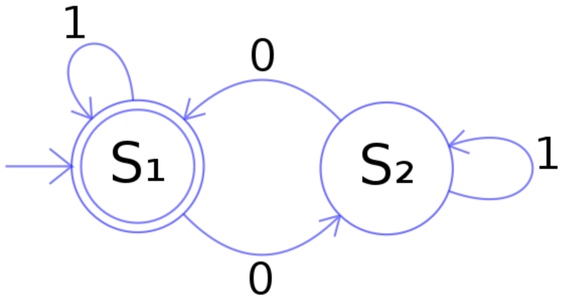
Constraining the Search Space with Input Grammars

Grammar

Formal Languages

Formal Language Descriptions

Formal Languages



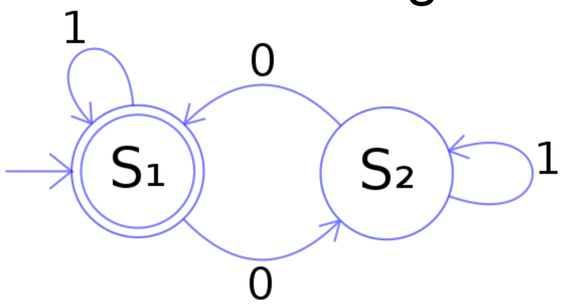
3. Regular

(Chomsky,1956)

Formal Language Descriptions

Formal Languages

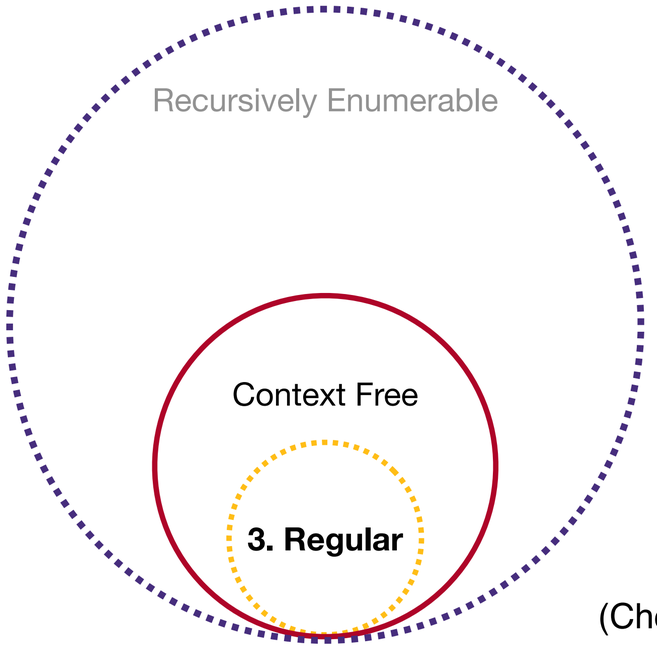
Argument Stack



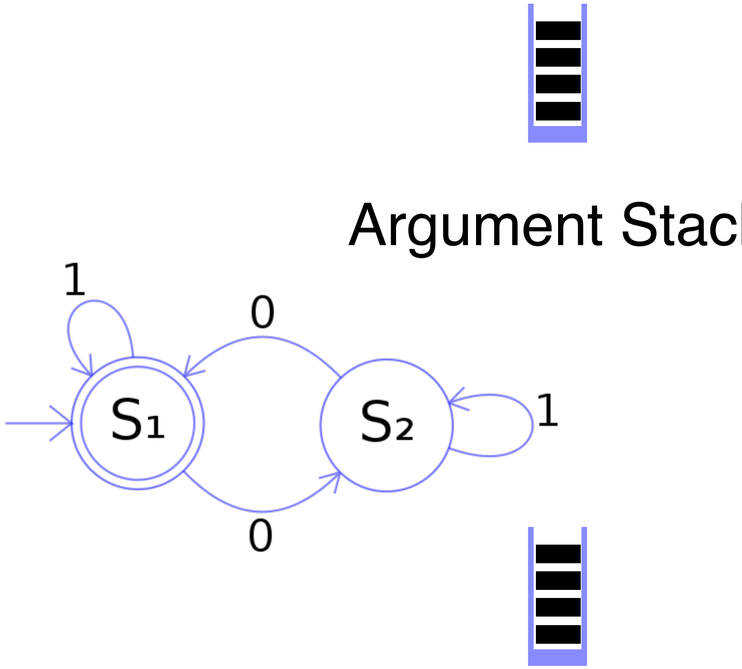
(Chomsky,1956)

Formal Language Descriptions

Formal Languages

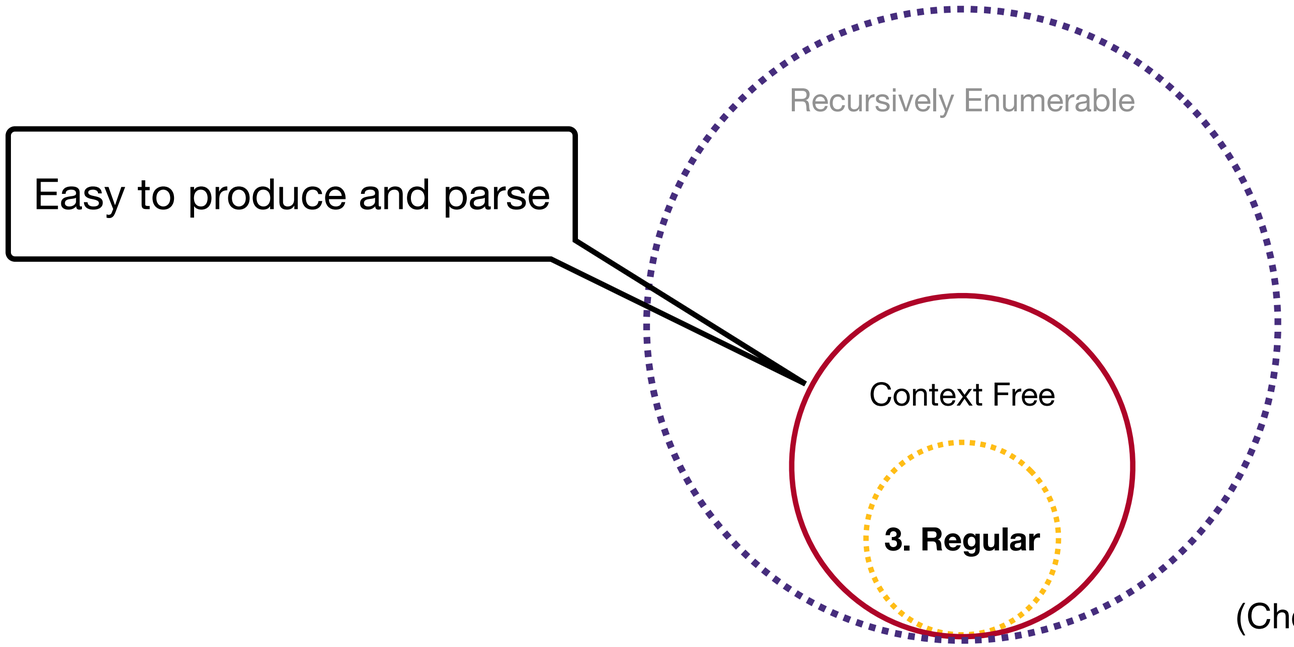


(Chomsky,1956)

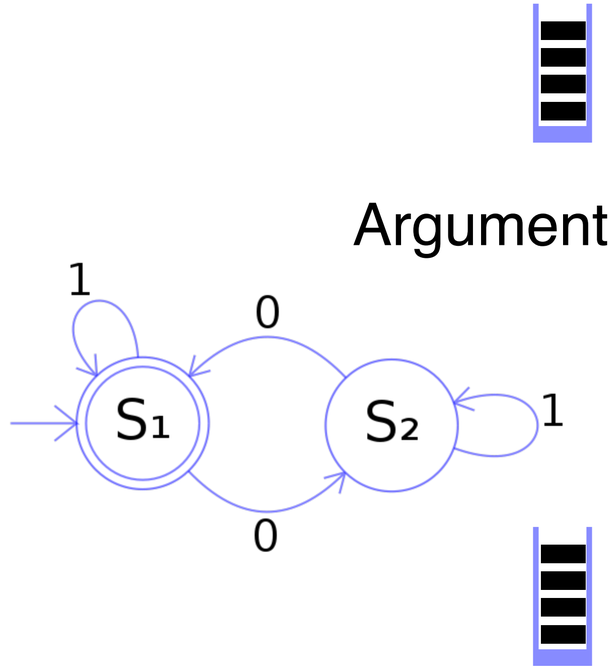


Formal Language Descriptions

Formal Languages



(Chomsky,1956)



Formal Language Descriptions

Grammar

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

Arithmetic expression grammar

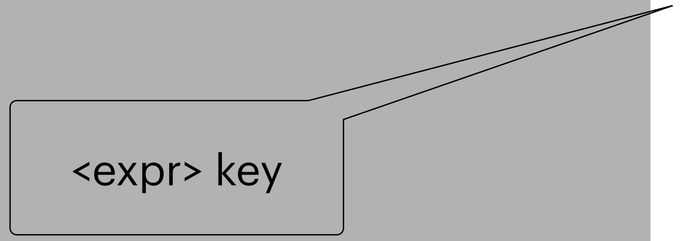
Grammar

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

Arithmetic expression grammar

Grammar

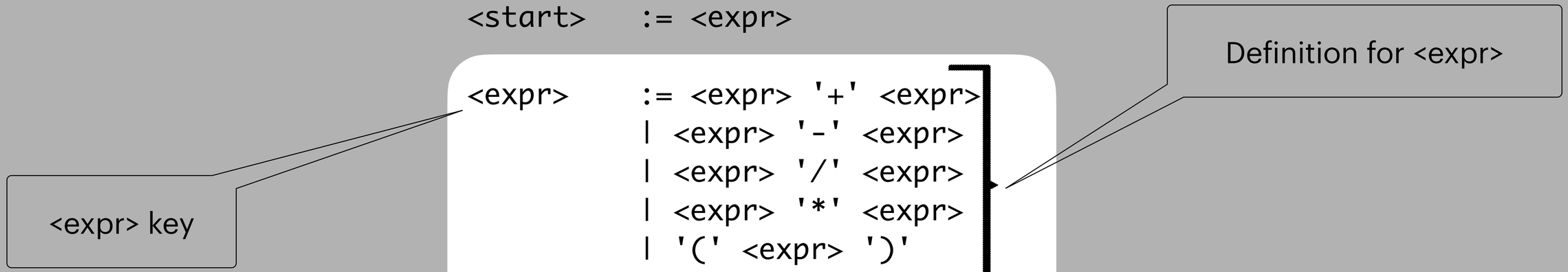
```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```



Arithmetic expression grammar

Grammar

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```



Arithmetic expression grammar

Grammar

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

Arithmetic expression grammar

Grammar

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

Expansion Rule

Arithmetic expression grammar

Grammar

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

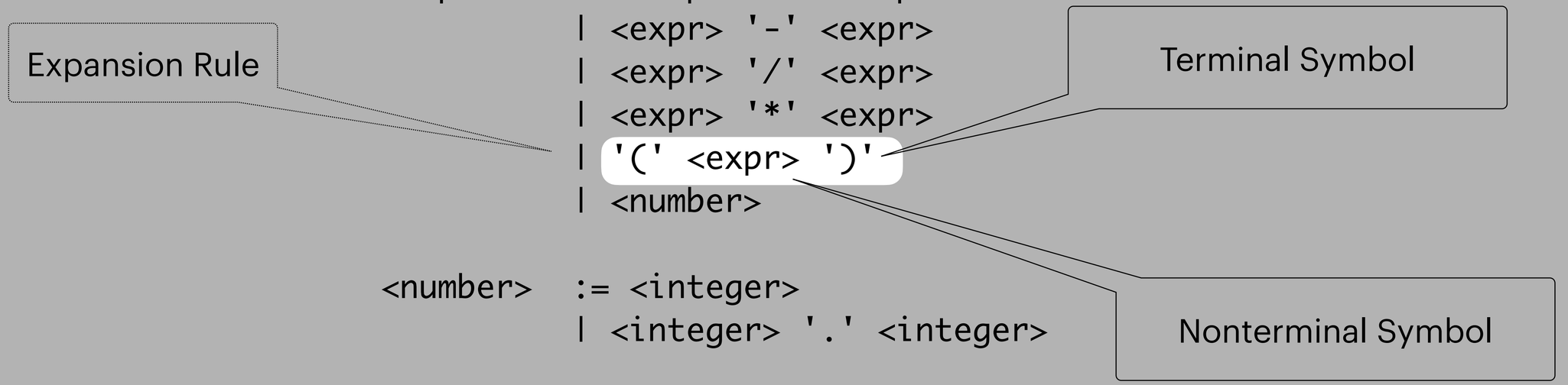
Expansion Rule

Terminal Symbol

Arithmetic expression grammar

Grammar

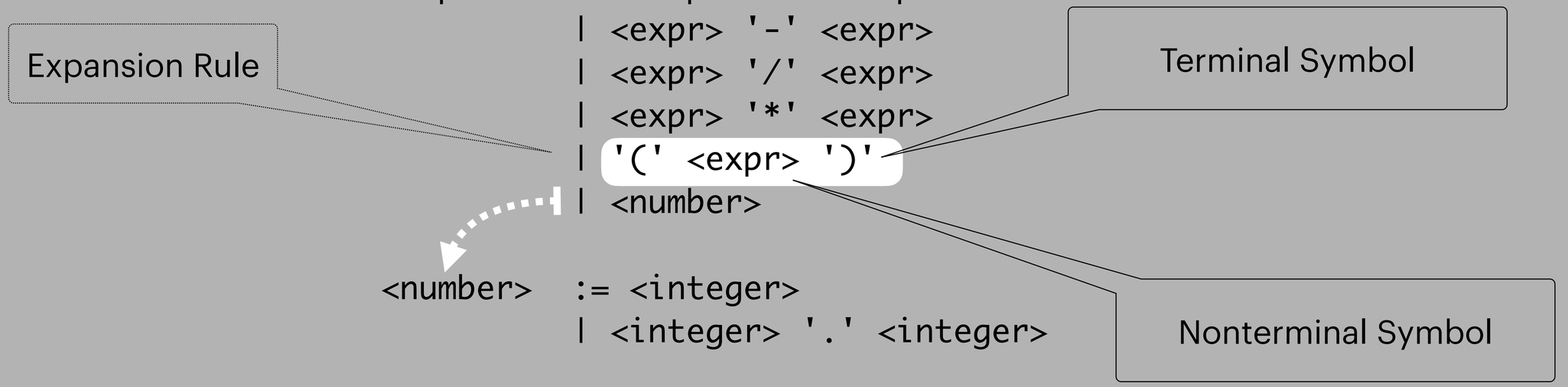
```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```



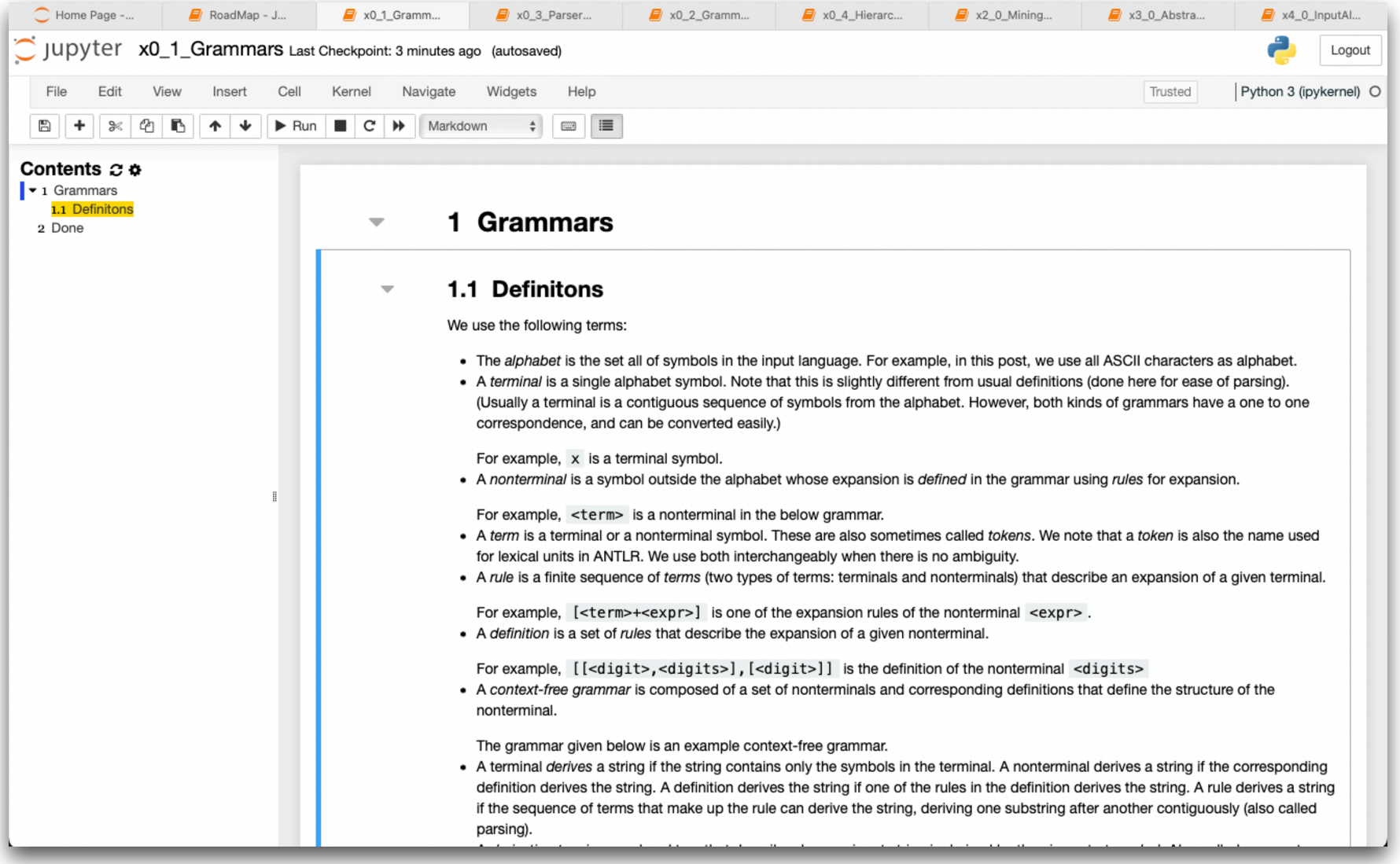
Arithmetic expression grammar

Grammar

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```



Arithmetic expression grammar



http://localhost:8888/notebooks/x0_1_Grammars.ipynb

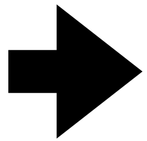
Grammars

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

For Parsing

Grammars

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

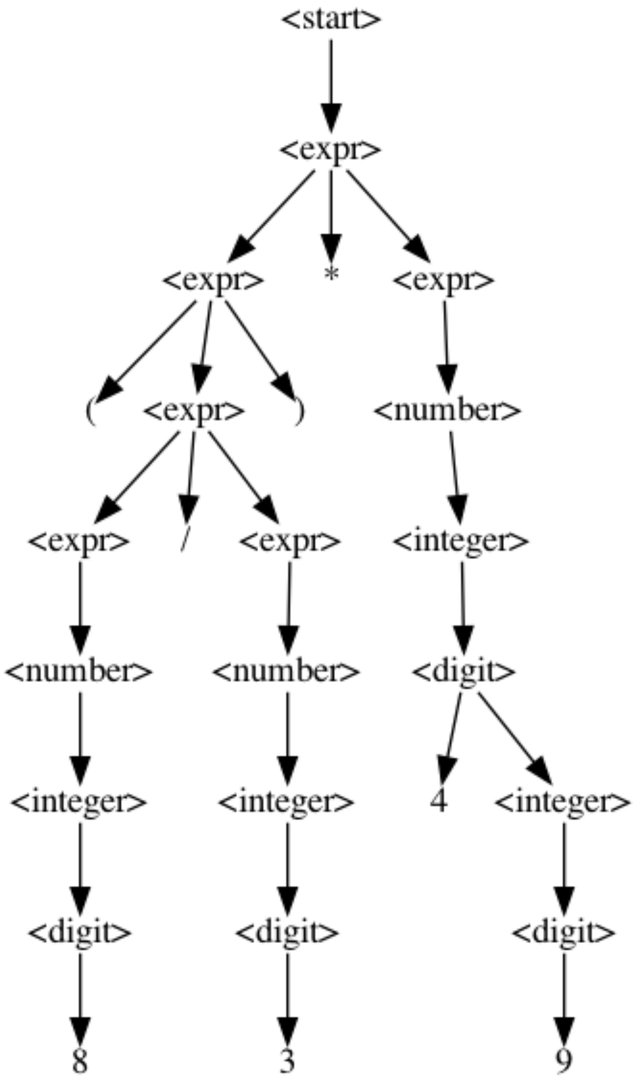
(8 / 3) * 49 

For Parsing

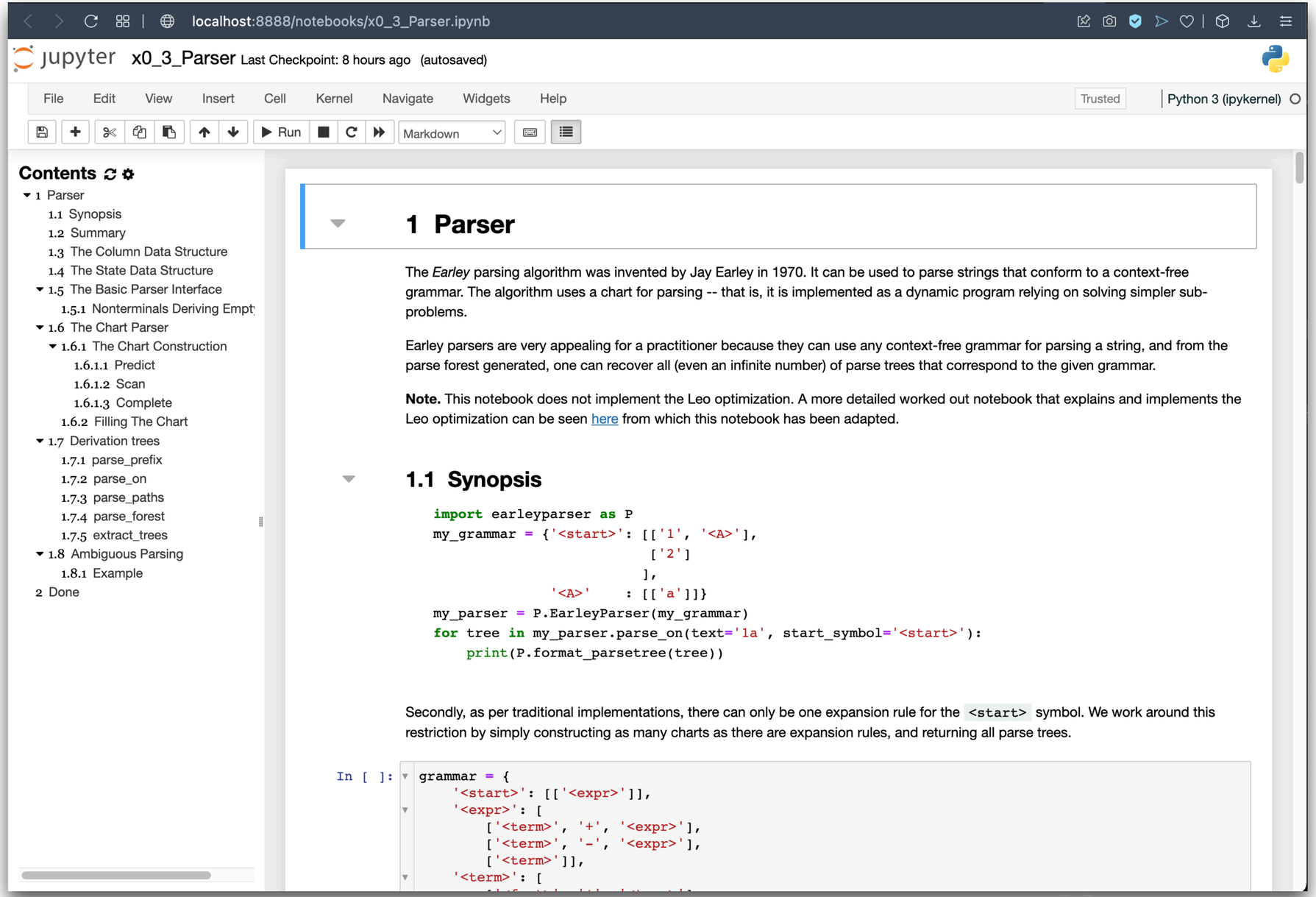
Grammars

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

(8 / 3) * 49



For Parsing



http://localhost:8888/notebooks/x0_3_Parser.ipynb

Grammars

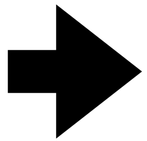
```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

For Fuzzing

(Hanford 1970)
(Purdom 1972)

Grammars

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```



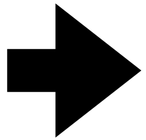
```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-
((-1 * +(8 - 5 - 6)) * (-(a-+(((+(4)
)))) - ++4) / +(-+---((5.6 - --(3 *
-1.8 * +(6 * +-(((-(-6) * ---+6)) /
+--(+-+-7 * (-0 * (+(((((2)) + 8 - 3
 - ++9.0 + ---(--+7 / (1 / +++6.37)
+ (1) / 482) / +++-+0)))) + 8.2 - 27
 - -9 / +((+9 * --2 + --+-+-((-1 * +
(8 - 5 - 6)) * (-(a-+(((+(4))))) - +
+4) / +(-+---((5.6 - --(3 * -1.8 * +
(6 * +-(((-(-6) * ---+6)) / +--(+-+-
7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0
 + ---(--+7 / (1 / +++6.37) + (1) /
482) / +++-+0)))) * -+5 + 7.513))))
- (+1 / ++((-84)))))))) * ++5 / +-(-
-2 - -++-9.0)))) / 5 * --++090 + * -
+5 + 7.513)))) - (+1 / ++((-84)))))))
)) * 8.2 - 27 - -9 / +((+9 * --2 + -
-+-+-((-1 * +(8 - 5 - 6)) * (-(a-+((
(+(4))))) - ++4) / +(-+---((5.6 - --
(3 * -1.8 * +(6 * +-(((-(-6) * ---+6
)) / +--(+-+-7 * (-0 * (+(((((2)) +
8 - 3 - ++9.0 + ---(--+7 / (1 / +++6
.37) + (1) / 482) / +++-+0)))) * -+5
 + 7.513)))) - (+1 / ++((-84)))))))
 * ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090 ++5 / +-(--2 - -++-9.0)))) /
 5 * --++090
```

For Fuzzing

(Hanford 1970)
(Purdom 1972)

```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-
((-1 * +(8 - 5 - 6)) * (-(a-+(((+(4)
)))) - ++4) / +(-+---((5.6 - --(3 *
-1.8 * +(6 * +-(((-(-6) * ---+6)) /
+--(+-+-7 * (-0 * (+(((((2)) + 8 - 3
 - ++9.0 + ---(--+7 / (1 / +++6.37)
+ (1) / 482) / +++-+0)))) + 8.2 - 27
 - -9 / +((+9 * --2 + --+-+-((-1 * +
(8 - 5 - 6)) * (-(a-+(((+(4))))) - +
+4) / +(-+---((5.6 - --(3 * -1.8 * +
(6 * +-(((-(-6) * ---+6)) / +--(+-+-
7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0
 + ---(--+7 / (1 / +++6.37) + (1) /
482) / +++-+0)))) * -+5 + 7.513))))
- (+1 / ++(((-84))))))) * ++5 / +-(-
-2 - -++-9.0)))) / 5 * --++090 + * -
+5 + 7.513)))) - (+1 / ++(((-84)))))
)) * 8.2 - 27 - -9 / +((+9 * --2 + -
-+-+-((-1 * +(8 - 5 - 6)) * (-(a-+((
(+(4))))) - ++4) / +(-+---((5.6 - --
(3 * -1.8 * +(6 * +-(((-(-6) * ---+6
)) / +--(+-+-7 * (-0 * (+(((((2)) +
8 - 3 - ++9.0 + ---(--+7 / (1 / +++6
.37) + (1) / 482) / +++-+0)))) * -+5
 + 7.513)))) - (+1 / ++(((-84)))))))
 * ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090 ++5 / +-(--2 - -++-9.0)))) /
 5 * --++090
```

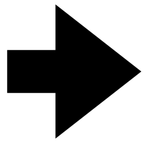
Grammars



As effective producers

```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-
((-1 * +(8 - 5 - 6)) * (-(a-+(((+(4)
)))) - ++4) / +(-+---((5.6 - --(3 *
-1.8 * +(6 * +-(((-(-6) * ---+6)) /
+--(+-+-7 * (-0 * (+(((((2)) + 8 - 3
 - ++9.0 + ---(--+7 / (1 / +++6.37)
+ (1) / 482) / +++-+0)))) + 8.2 - 27
 - -9 / +((+9 * --2 + --+-+-((-1 * +
(8 - 5 - 6)) * (-(a-+(((+(4))))) - +
+4) / +(-+---((5.6 - --(3 * -1.8 * +
(6 * +-(((-(-6) * ---+6)) / +--(+-+-
7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0
 + ---(--+7 / (1 / +++6.37) + (1) /
482) / +++-+0)))) * -+5 + 7.513))))
- (+1 / ++(((-84))))))) * ++5 / +-(-
-2 - -++-9.0)))) / 5 * --++090 + * -
+5 + 7.513)))) - (+1 / ++(((-84)))))
)) * 8.2 - 27 - -9 / +((+9 * --2 + -
-+-+-((-1 * +(8 - 5 - 6)) * (-(a-+((
(+(4))))) - ++4) / +(-+---((5.6 - --
(3 * -1.8 * +(6 * +-(((-(-6) * ---+6
)) / +--(+-+-7 * (-0 * (+(((((2)) +
8 - 3 - ++9.0 + ---(--+7 / (1 / +++6
.37) + (1) / 482) / +++-+0)))) * -+5
 + 7.513)))) - (+1 / ++(((-84)))))))
 * ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090 ++5 / +-(--2 - -++-9.0)))) /
 5 * --++090
```

Grammars



As effective producers

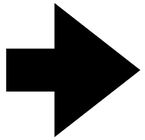
Grammars

```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```

As efficient producers

Grammars

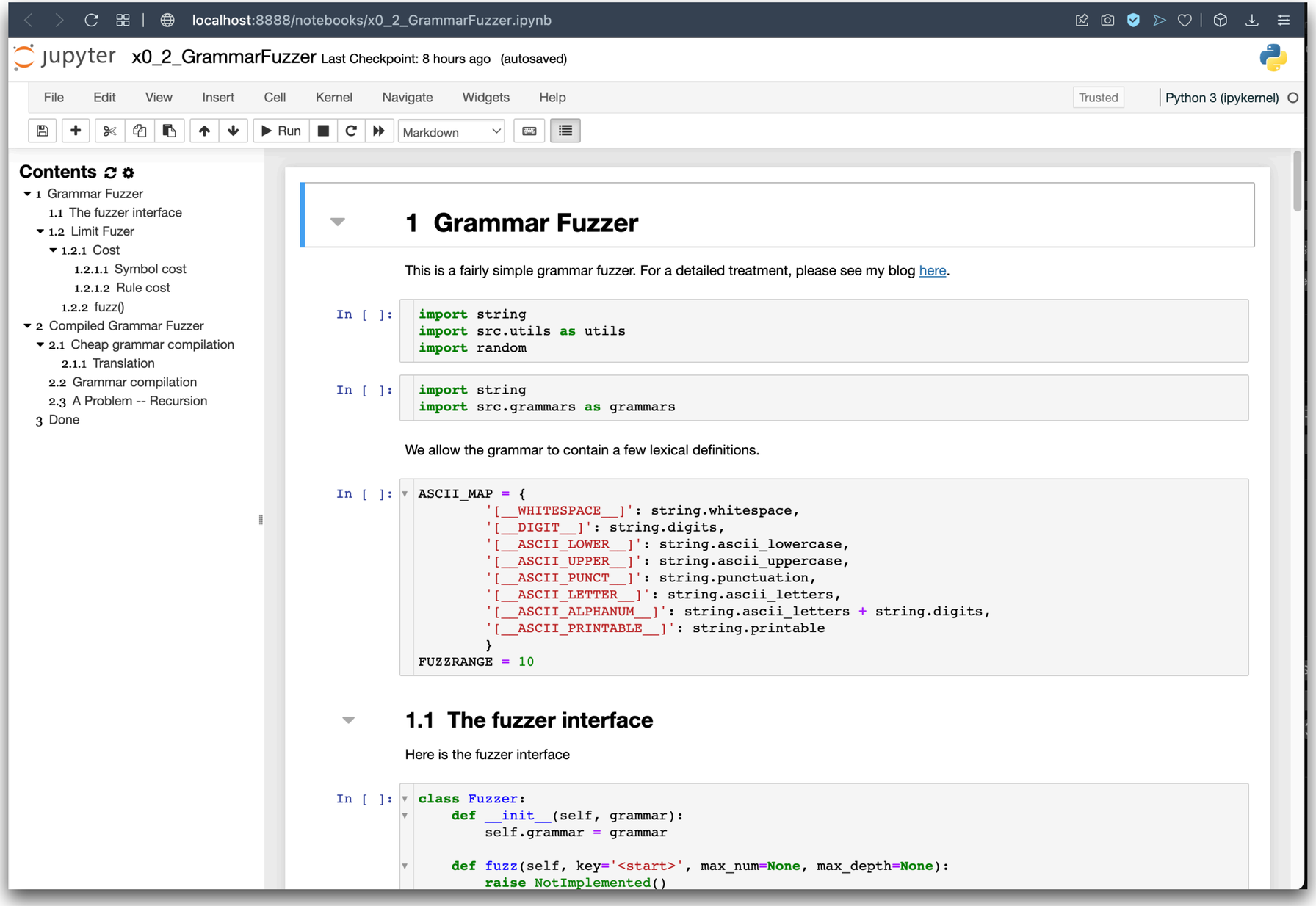
```
<start>   := <expr>

<expr>    := <expr> '+' <expr>
           | <expr> '-' <expr>
           | <expr> '/' <expr>
           | <expr> '*' <expr>
           | '(' <expr> ')'
           | <number>

<number>  := <integer>
           | <integer> '.' <integer>

<integer>:= <digit> <integer>
           | <digit>

<digit>  := [0-9]
```



Compiled Grammar (F1)

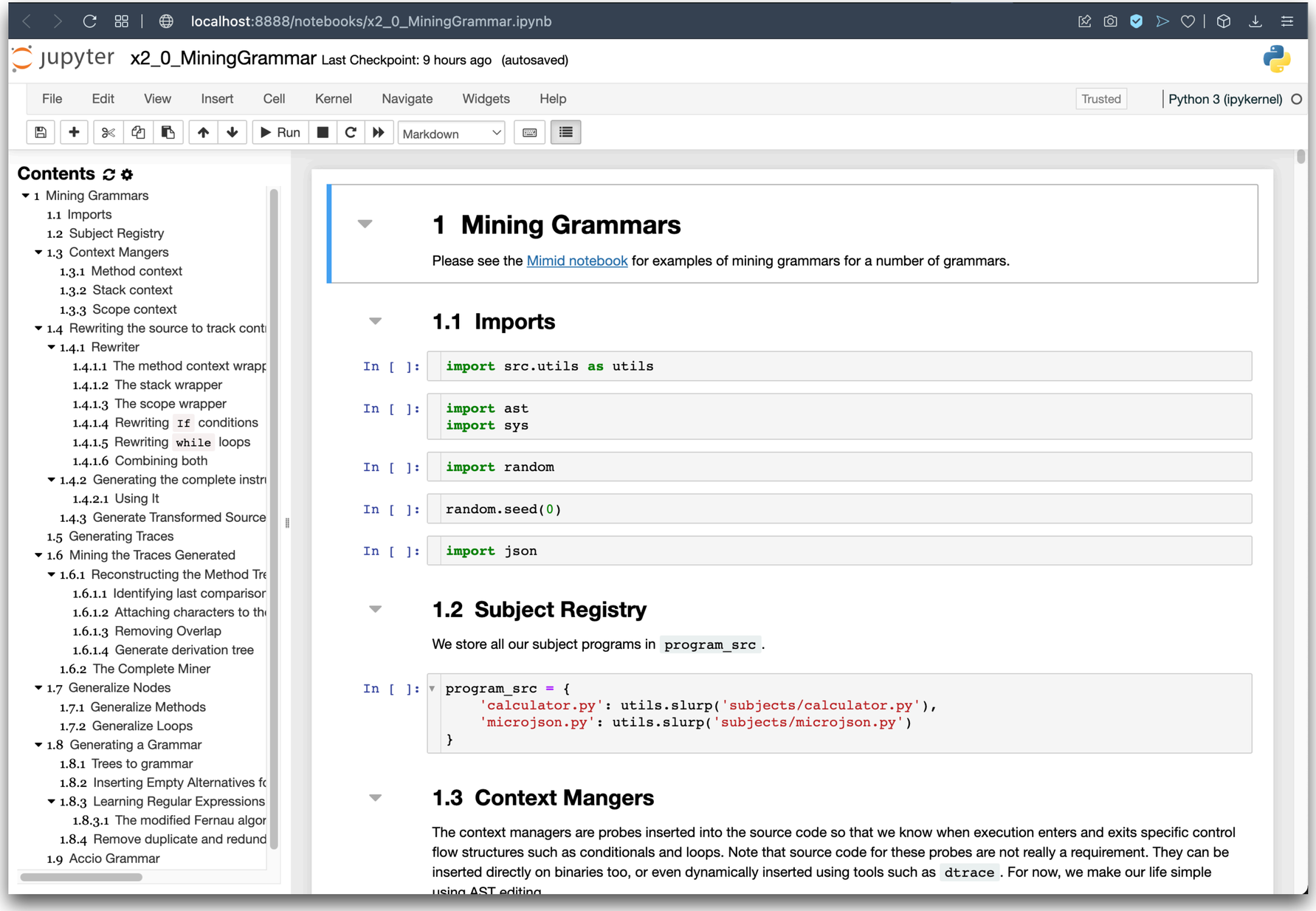
```
def start():
    expr()

def expr():
    match (random() % 6):
        case 0: expr(); print('+'); expr()
        case 1: expr(); print('-'); expr()
        case 2: expr(); print('/'); expr()
        case 3: expr(); print('*'); expr()
        case 4: print('('); expr(); print(')')
        case 5: number()

def number():
    match (random() % 2):
        case 0: integer()
        case 1: integer(); print('.'); integer()

def integer():
    match (random() % 2):
        case 0: digit(); integer()
        case 1: digit()

def digit():
    match (random() % 10):
        case 0: print('0')
        case 1: print('1')
        case 2: print('2')
        case 3: print('3')
        case 4: print('4')
        case 5: print('5')
        case 6: print('6')
        case 7: print('7')
```

As efficient producers



http://localhost:8888/notebooks/x0_2_GrammarFuzzer.ipynb

Where to Get the Input Grammar From?



http://localhost:8888/notebooks/x2_0_MiningGrammar.ipynb

Where to Get the Grammar From?

Hand-written parsers already encode the grammar

Where to Get the Grammar From?

Hand-written parsers already encode the grammar

1. Extract the input string accesses
2. Attach control flow information (context-managers)

How to Extract This Grammar?

How to Extract This Grammar?

- Inputs + control flow -> Dynamic Control Dependence Trees

How to Extract This Grammar?

- Inputs + control flow -> Dynamic Control Dependence Trees
- DCD Trees -> Parse Tree

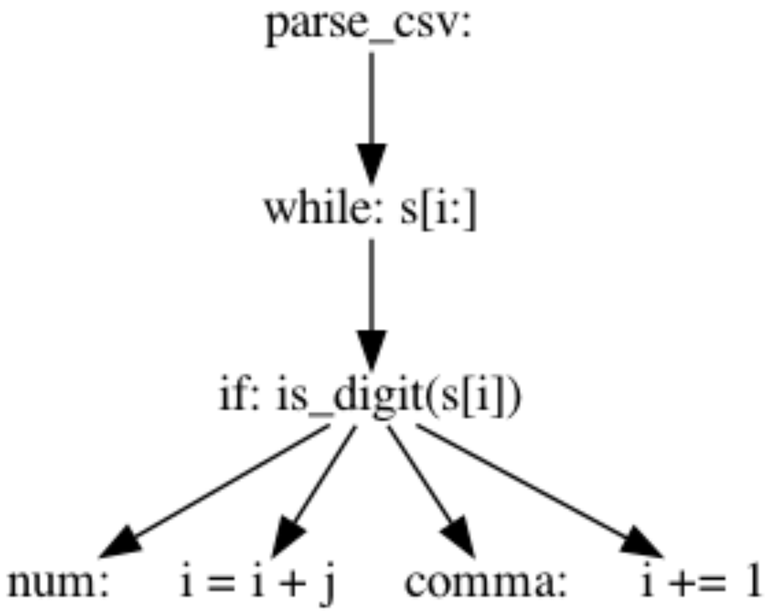
Control Dependence Graph

```
def parse_csv(s,i):
  while s[i:]:
    if is_digit(s[i]):
     n,j = num(s[i:])
     i = i+j
    else:
     comma(s[i])
     i += 1
```

Statement **B** is control dependent on **A** if A determines whether B executes.

Control Dependence Graph

```
def parse_csv(s,i):
  while s[i:]:
    if is_digit(s[i]):
      n,j = num(s[i:])
      i = i+j
    else:
      comma(s[i])
      i += 1
```

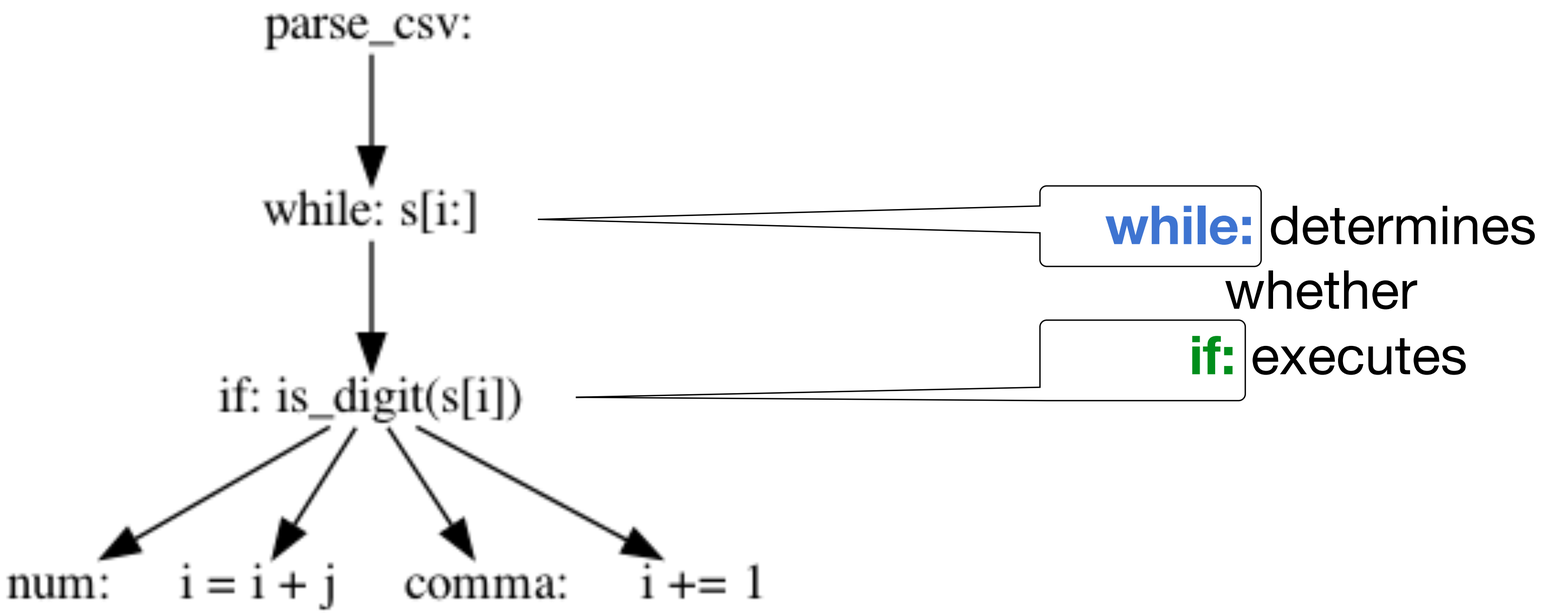


CDG for parse_csv

Statement **B** is control dependent on **A** if A determines whether B executes.

Control Dependence Graph

```
def parse_csv(s,i):
  while s[i:]:
    if is_digit(s[i]):
      n,j = num(s[i:])
      i = i+j
    else:
      comma(s[i])
      i += 1
```

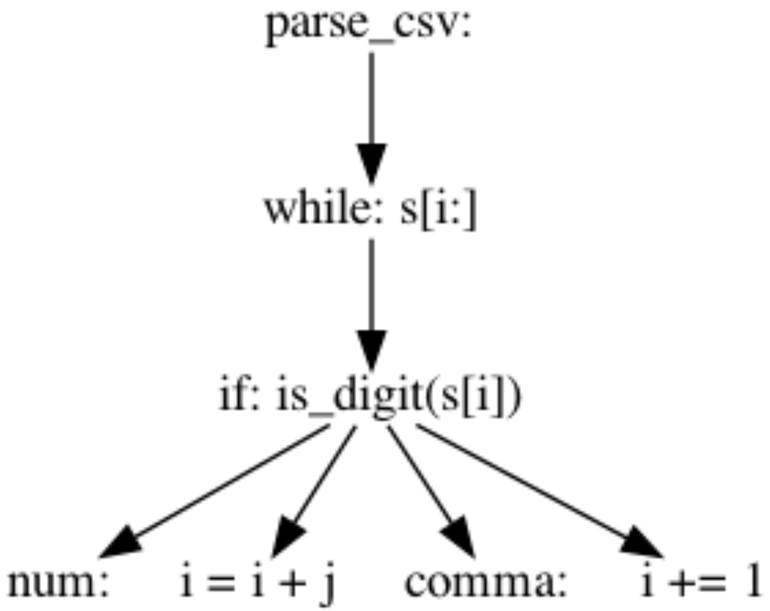


CDG for parse_csv

Statement **B** is control dependent on **A** if A determines whether B executes.

Dynamic Control Dependence Tree

```
def parse_csv(s,i):
  while s[i:]:
    if is_digit(s[i]):
      n,j = num(s[i:])
      i = i+j
    else:
      comma(s[i])
      i += 1
```

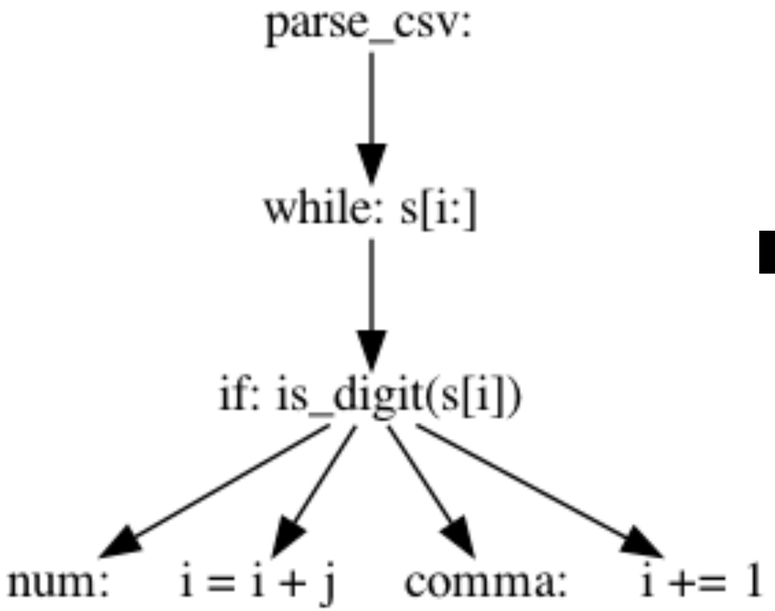


CDG for parse_csv

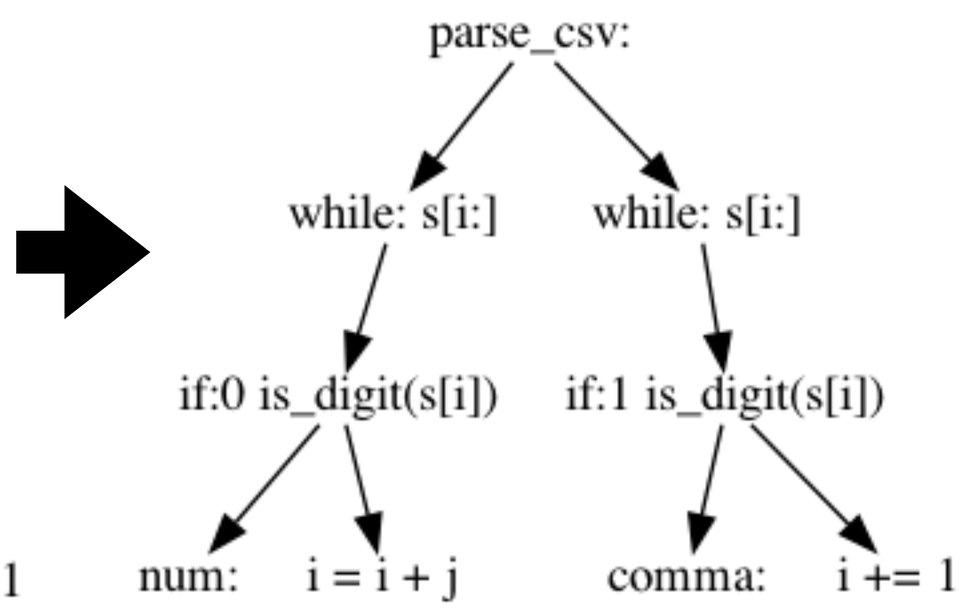
Each statement execution is represented as a separate node

Dynamic Control Dependence Tree

```
def parse_csv(s,i):
  while s[i:]:
    if is_digit(s[i]):
      n,j = num(s[i:])
      i = i+j
    else:
      comma(s[i])
      i += 1
```



CDG for parse_csv

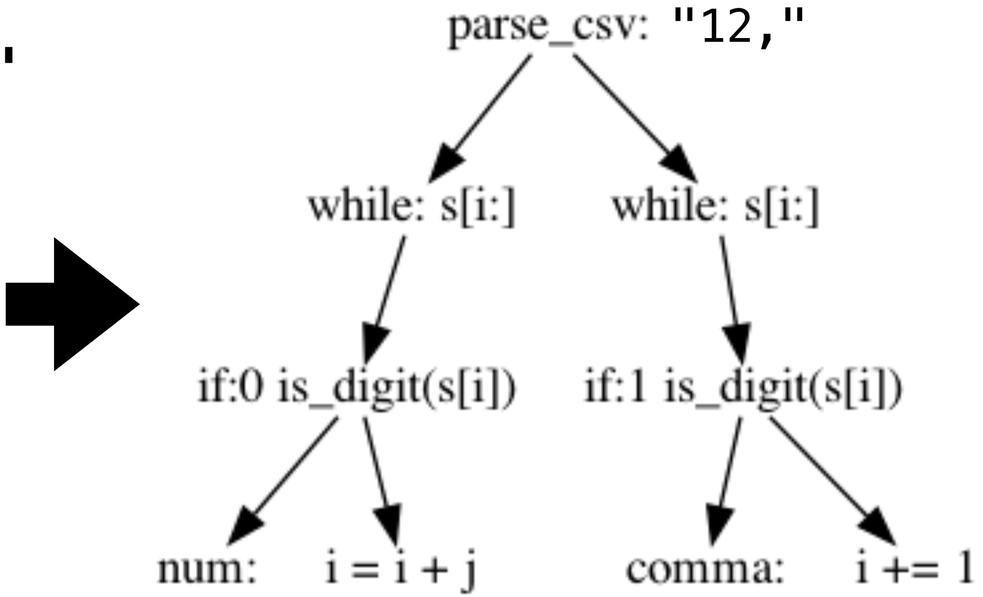


DCD Tree for call parse_csv()

Each statement execution is represented as a separate node

DCD Tree ~ Parse Tree

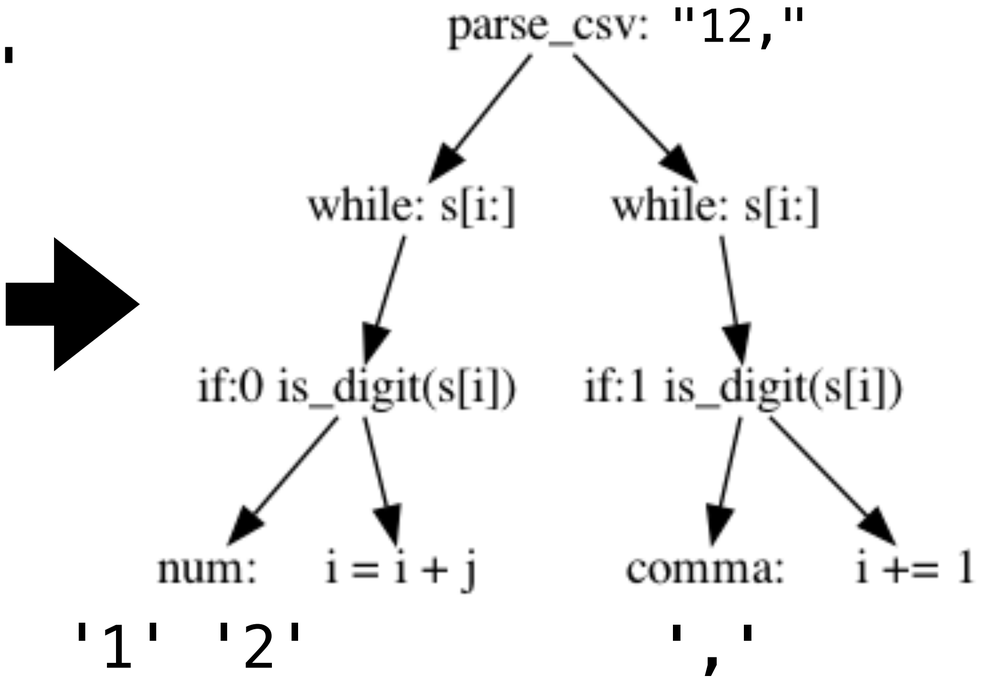
```
def parse_csv(s,i):  "12,"
  while s[i:]:
    if is_digit(s[i]):
      n,j = num(s[i:])
      i = i+j
    else:
      comma(s[i])
      i += 1
```



- No tracking beyond input buffer
- Characters are attached to nodes where they are **accessed last**

DCD Tree ~ Parse Tree

```
def parse_csv(s,i):  "12,"
  while s[i:]:
    if is_digit(s[i]):
     n,j = num(s[i:])
     i = i+j
    else:
     comma(s[i])
     i += 1
```



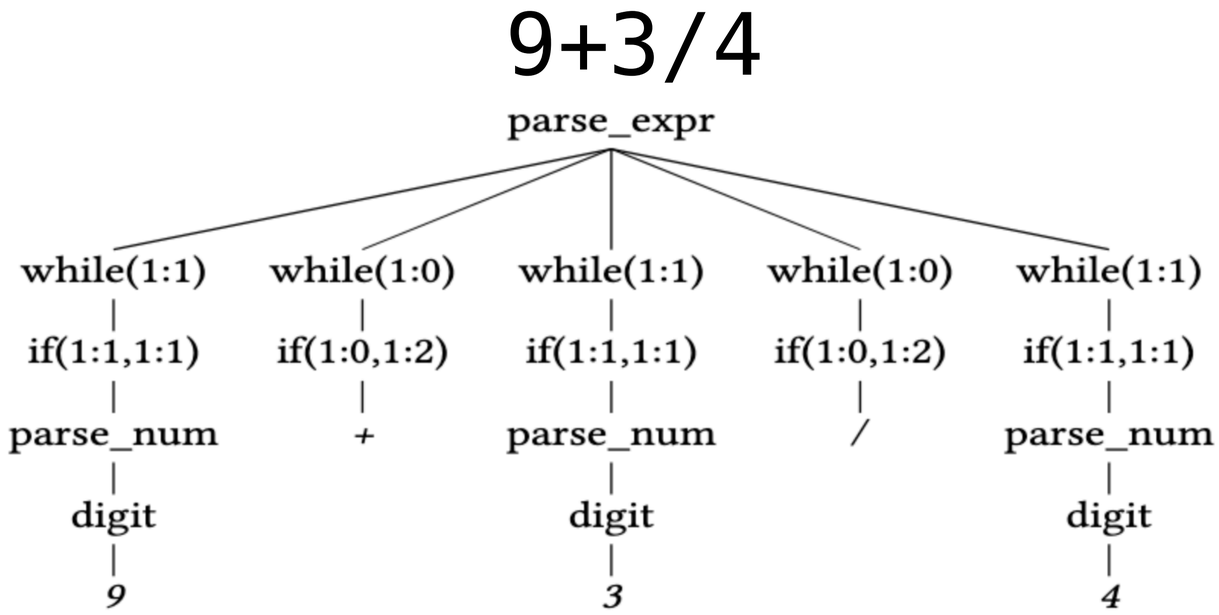
- No tracking beyond input buffer
- Characters are attached to nodes where they are **accessed last**

```
def is_digit(i): return i in '0123456789'
def parse_num(s,i):
    n = ''
    while s[i:] and is_digit(s[i]):
        n += s[i]
        i = i +1
    return i,n

def parse_paren(s, i):
    assert s[i] == '('
    i, v = parse_expr(s, i+1)
    if s[i:] == '': raise Ex(s, i)
    assert s[i] == ')'
    return i+1, v

def parse_expr(s, i = 0):
    expr, is_op = [], True
    while s[i:]:
        c = s[i]
        if isdigit(c):
            if not is_op: raise Ex(s,i)
            i,num = parse_num(s,i)
            expr.append(num)
            is_op = False
        elif c in ['+', '-', '*', '/']:
            if is_op: raise Ex(s,i)
            expr.append(c)
            is_op, i = True, i + 1
        elif c == '(':
            if not is_op: raise Ex(s,i)
            i, cexpr = parse_paren(s, i)
            expr.append(cexpr)
            is_op = False
        elif c == ')': break
        else: raise Ex(s,i)
    if is_op: raise Ex(s,i)
    return i, expr
```

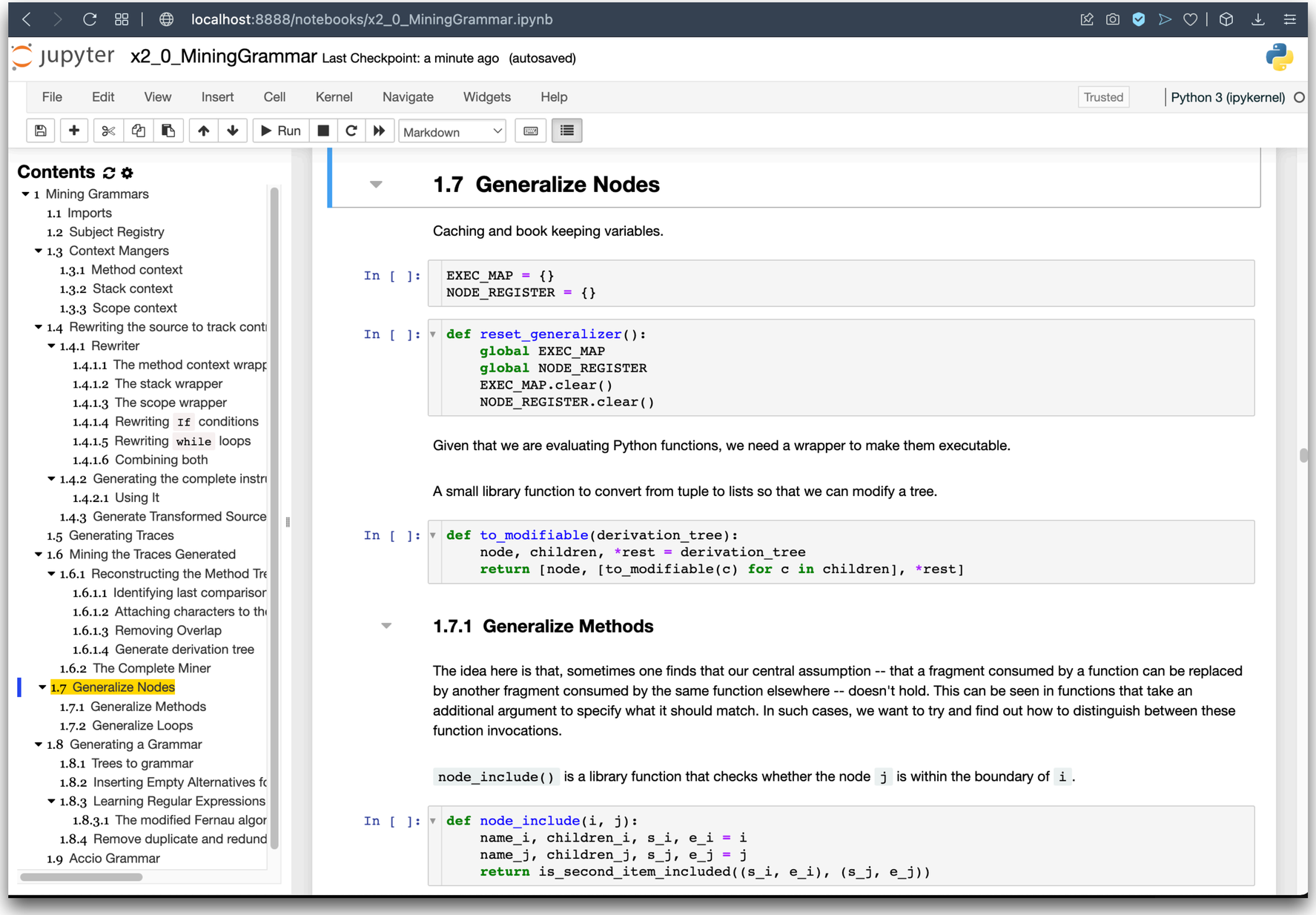
9+3/4

Parse tree for **parse_expr('9+3/4')**

```
def is_digit(i): return i in '0123456789'
def parse_num(s,i):
    n = ''
    while s[i:] and is_digit(s[i]):
        n += s[i]
        i = i +1
    return i,n

def parse_paren(s, i):
    assert s[i] == '('
    i, v = parse_expr(s, i+1)
    if s[i:] == '': raise Ex(s, i)
    assert s[i] == ')'
    return i+1, v

def parse_expr(s, i = 0):
    expr, is_op = [], True
    while s[i:]:
        c = s[i]
        if isdigit(c):
            if not is_op: raise Ex(s,i)
            i,num = parse_num(s,i)
            expr.append(num)
            is_op = False
        elif c in ['+', '-', '*', '/']:
            if is_op: raise Ex(s,i)
            expr.append(c)
            is_op, i = True, i + 1
        elif c == '(':
            if not is_op: raise Ex(s,i)
            i, cexpr = parse_paren(s, i)
            expr.append(cexpr)
            is_op = False
        elif c == ')': break
        else: raise Ex(s,i)
    if is_op: raise Ex(s,i)
    return i, expr
```

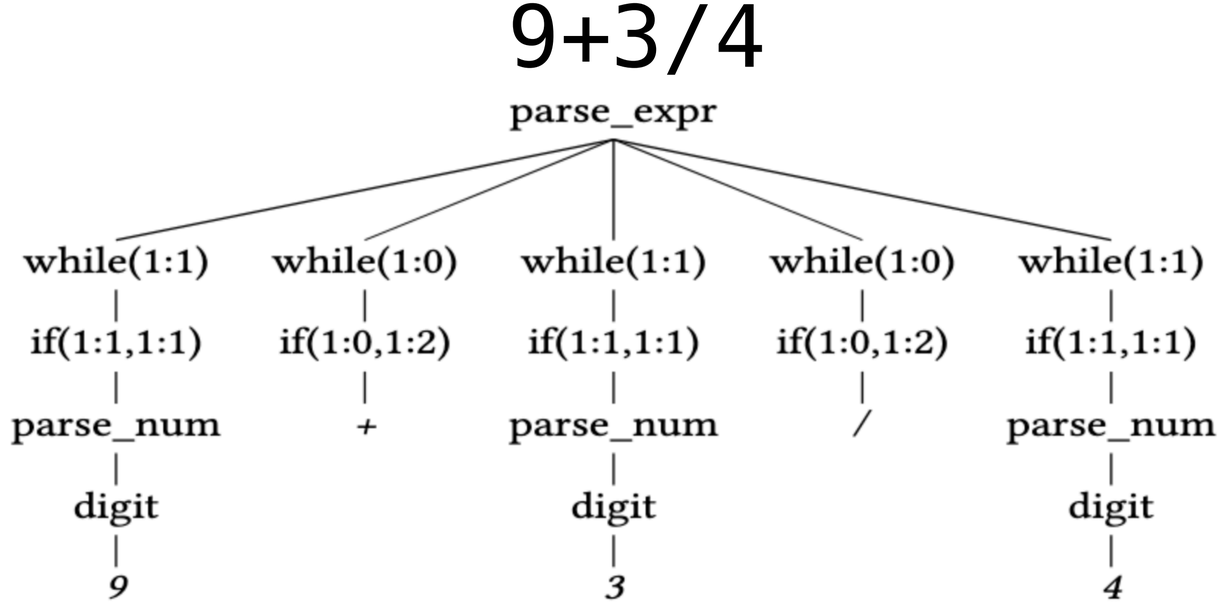


Parse tree for **parse_expr('9+3/4')**


localhost:8888/notebooks/x2_0_MiningGrammar.ipynb
jupyter x2_0_MiningGrammar Last Checkpoint: a minute ago (autosaved)
File Edit View Insert Cell Kernel Navigate Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
Contents
1 Mining Grammars
1.1 Imports
1.2 Subject Registry
1.3 Context Mangers
1.3.1 Method context
1.3.2 Stack context
1.3.3 Scope context
1.4 Rewriting the source to track conti
1.4.1 Rewriter
1.4.1.1 The method context wrapp
1.4.1.2 The stack wrapper
1.4.1.3 The scope wrapper
1.4.1.4 Rewriting If conditions
1.4.1.5 Rewriting while loops
1.4.1.6 Combining both
1.4.2 Generating the complete instr
1.4.2.1 Using It
1.4.3 Generate Transformed Source
1.5 Generating Traces
1.6 Mining the Traces Generated
1.6.1 Reconstructing the Method Tre
1.6.1.1 Identifying last comparisor
1.6.1.2 Attaching characters to th
1.6.1.3 Removing Overlap
1.6.1.4 Generate derivation tree
1.6.2 The Complete Miner
1.7 Generalize Nodes
1.7.1 Generalize Methods
1.7.2 Generalize Loops
1.8 Generating a Grammar
1.8.1 Trees to grammar
1.8.2 Inserting Empty Alternatives fo
1.8.3 Learning Regular Expressions
1.8.3.1 The modified Fernau algor
1.8.4 Remove duplicate and redund
1.9 Accio Grammar
1.7 Generalize Nodes
Caching and book keeping variables.
In [ ]:
EXEC_MAP = {}
NODE_REGISTER = {}
In [ ]:
def reset_generalizer():
global EXEC_MAP
global NODE_REGISTER
EXEC_MAP.clear()
NODE_REGISTER.clear()
Given that we are evaluating Python functions, we need a wrapper to make them executable.
A small library function to convert from tuple to lists so that we can modify a tree.
In [ ]:
def to_modifiable(derivation_tree):
node, children, *rest = derivation_tree
return [node, [to_modifiable(c) for c in children], *rest]
1.7.1 Generalize Methods
The idea here is that, sometimes one finds that our central assumption -- that a fragment consumed by a function can be replaced by another fragment consumed by the same function elsewhere -- doesn't hold. This can be seen in functions that take an additional argument to specify what it should match. In such cases, we want to try and find out how to distinguish between these function invocations.
node_include() is a library function that checks whether the node j is within the boundary of i .
In [ ]:
def node_include(i, j):
name_i, children_i, s_i, e_i = i
name_j, children_j, s_j, e_j = j
return is_second_item_included((s_i, e_i), (s_j, e_j))

```
def is_digit(i): return i in '0123456789'
def parse_num(s,i):
    n = ''
    while s[i:] and is_digit(s[i]):
        n += s[i]
        i = i +1
    return i,n

def parse_paren(s, i):
    assert s[i] == '('
    i, v = parse_expr(s, i+1)
    if s[i:] == '': raise Ex(s, i)
    assert s[i] == ')'
    return i+1, v

def parse_expr(s, i = 0):
    expr, is_op = [], True
    while s[i:]:
        c = s[i]
        if isdigit(c):
            if not is_op: raise Ex(s,i)
            i,num = parse_num(s,i)
            expr.append(num)
            is_op = False
        elif c in ['+', '-', '*', '/']:
            if is_op: raise Ex(s,i)
            expr.append(c)
            is_op, i = True, i + 1
        elif c == '(':
            if not is_op: raise Ex(s,i)
            i, cexpr = parse_paren(s, i)
            expr.append(cexpr)
            is_op = False
        elif c == ')': break
        else: raise Ex(s,i)
    if is_op: raise Ex(s,i)
    return i, expr
```

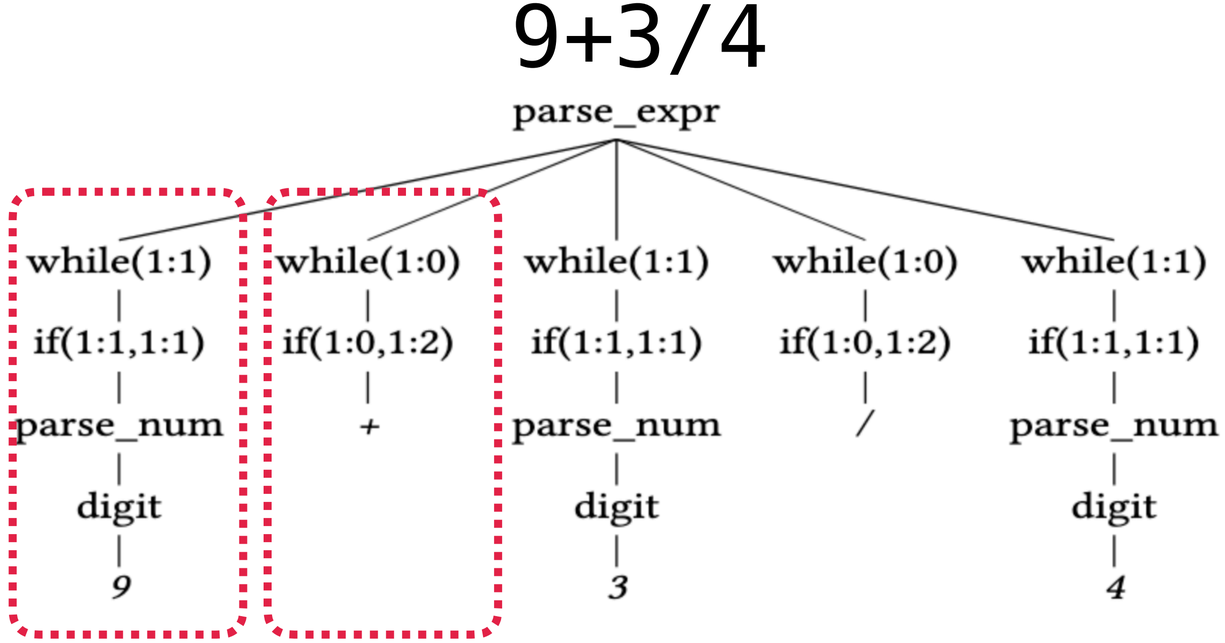
Identifying Compatible Nodes



Which nodes correspond to the same nonterminal

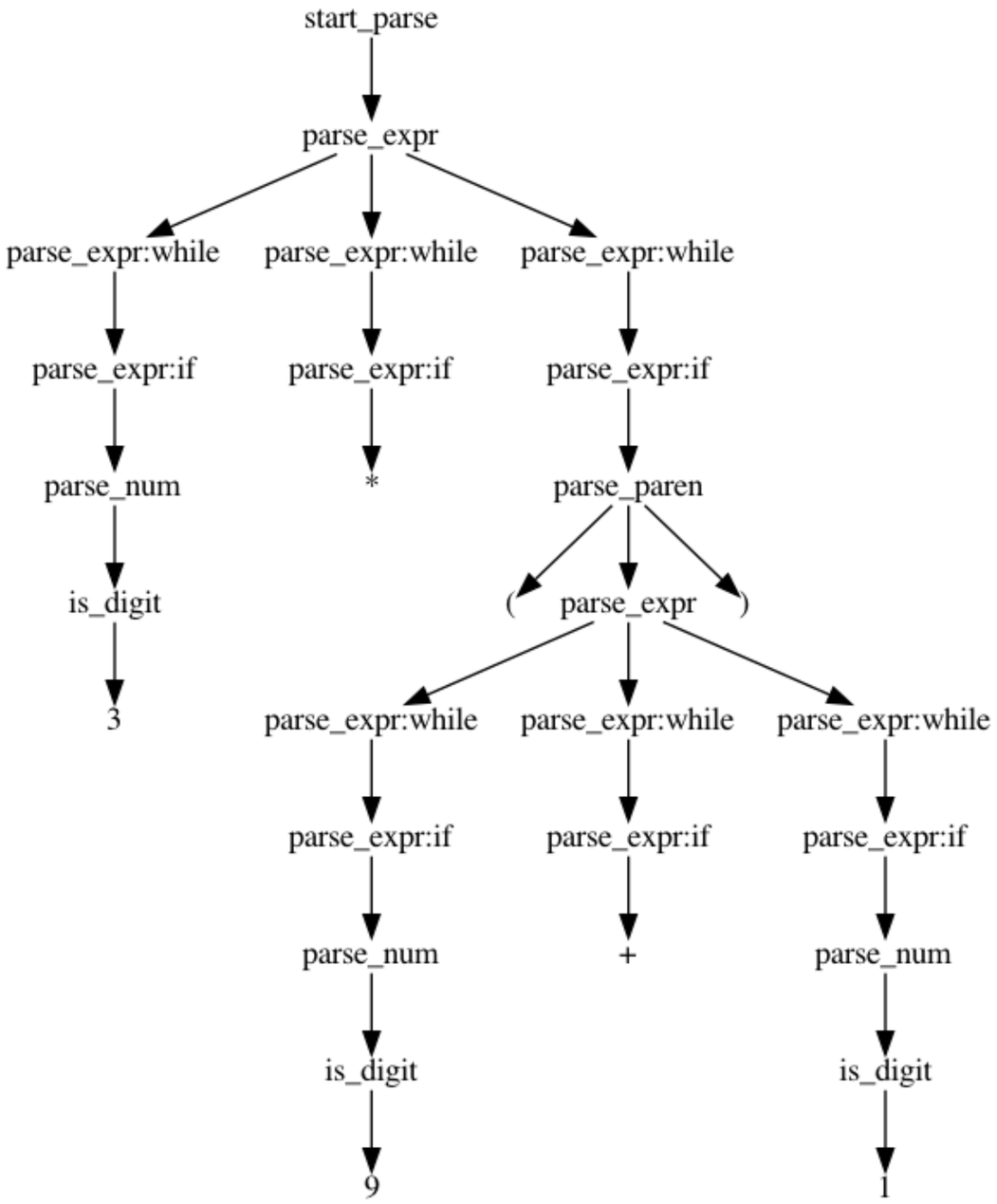
```
def is_digit(i): return i in '0123456789'
def parse_num(s,i):
    n = ''
    while s[i:] and is_digit(s[i]):
        n += s[i]
        i = i +1
    return i,n

def parse_paren(s, i):
    assert s[i] == '('
    i, v = parse_expr(s, i+1)
    if s[i:] == '': raise Ex(s, i)
    assert s[i] == ')'
    return i+1, v

def parse_expr(s, i = 0):
    expr, is_op = [], True
    while s[i:]:
        c = s[i]
        if isdigit(c):
            if not is_op: raise Ex(s,i)
            i,num = parse_num(s,i)
            expr.append(num)
            is_op = False
        elif c in ['+', '-', '*', '/']:
            if is_op: raise Ex(s,i)
            expr.append(c)
            is_op, i = True, i + 1
        elif c == '(':
            if not is_op: raise Ex(s,i)
            i, cexpr = parse_paren(s, i)
            expr.append(cexpr)
            is_op = False
        elif c == ')': break
        else: raise Ex(s,i)
    if is_op: raise Ex(s,i)
    return i, expr
```

Identifying Compatible Nodes

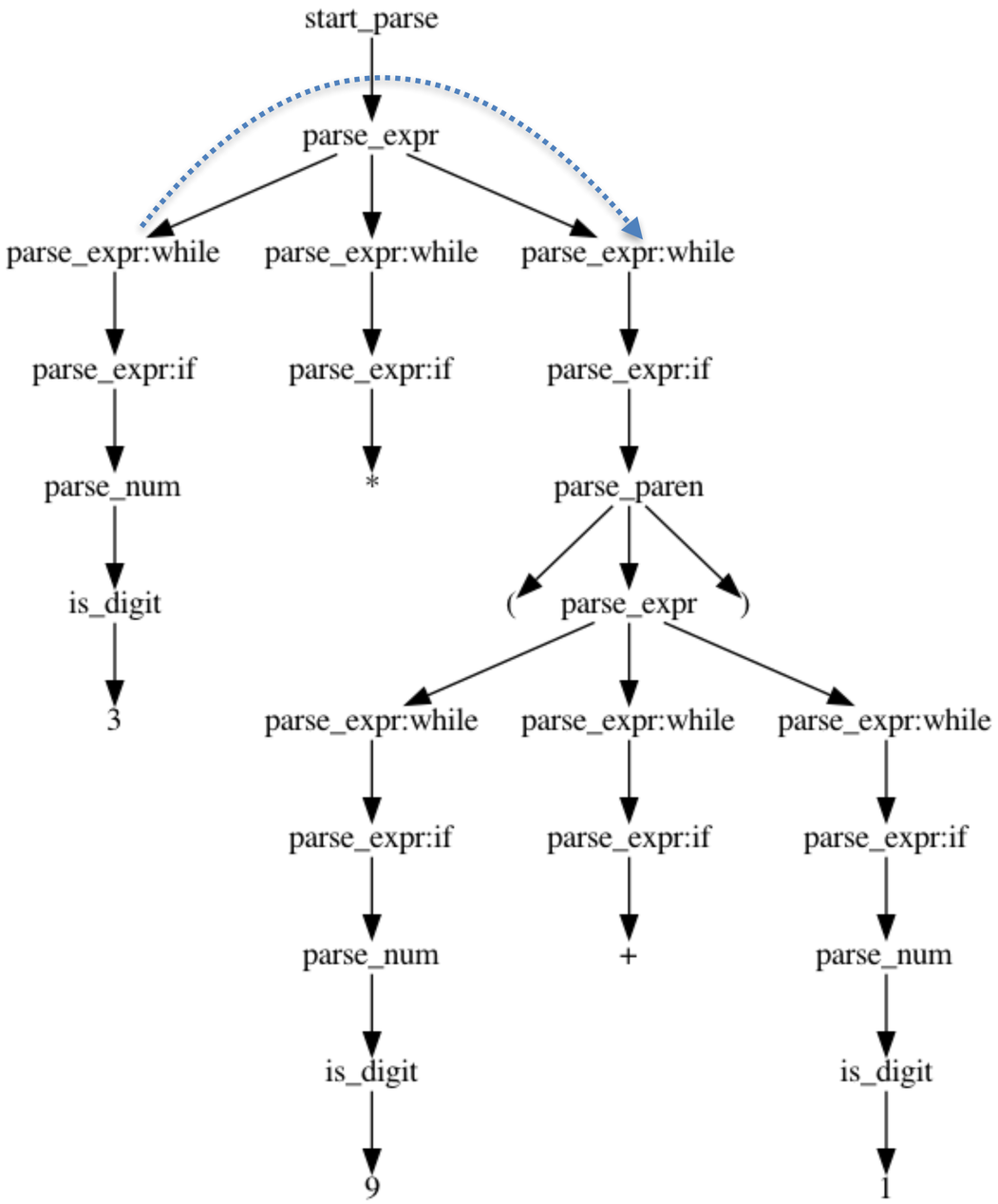


Which nodes correspond to the same nonterminal

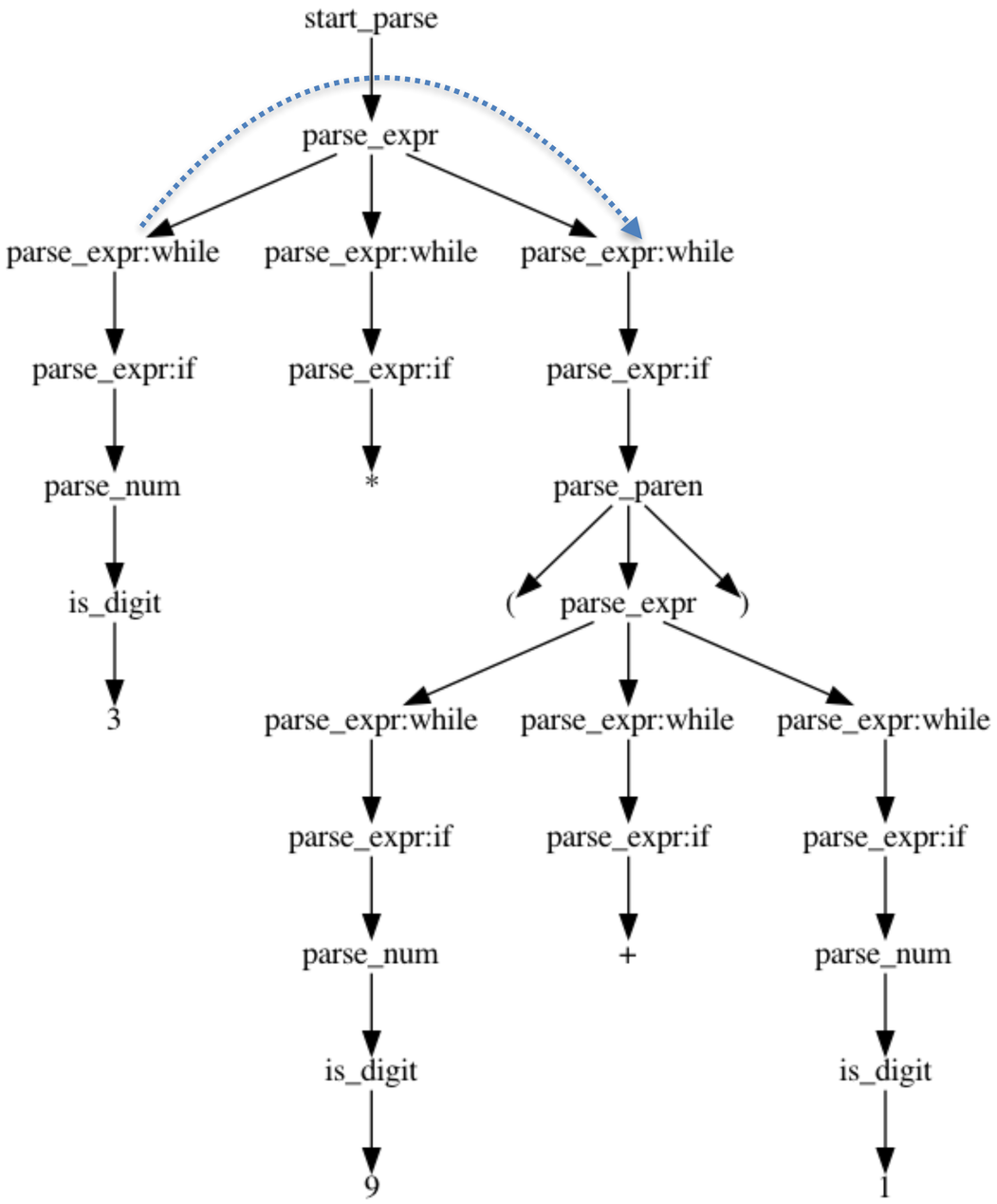
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

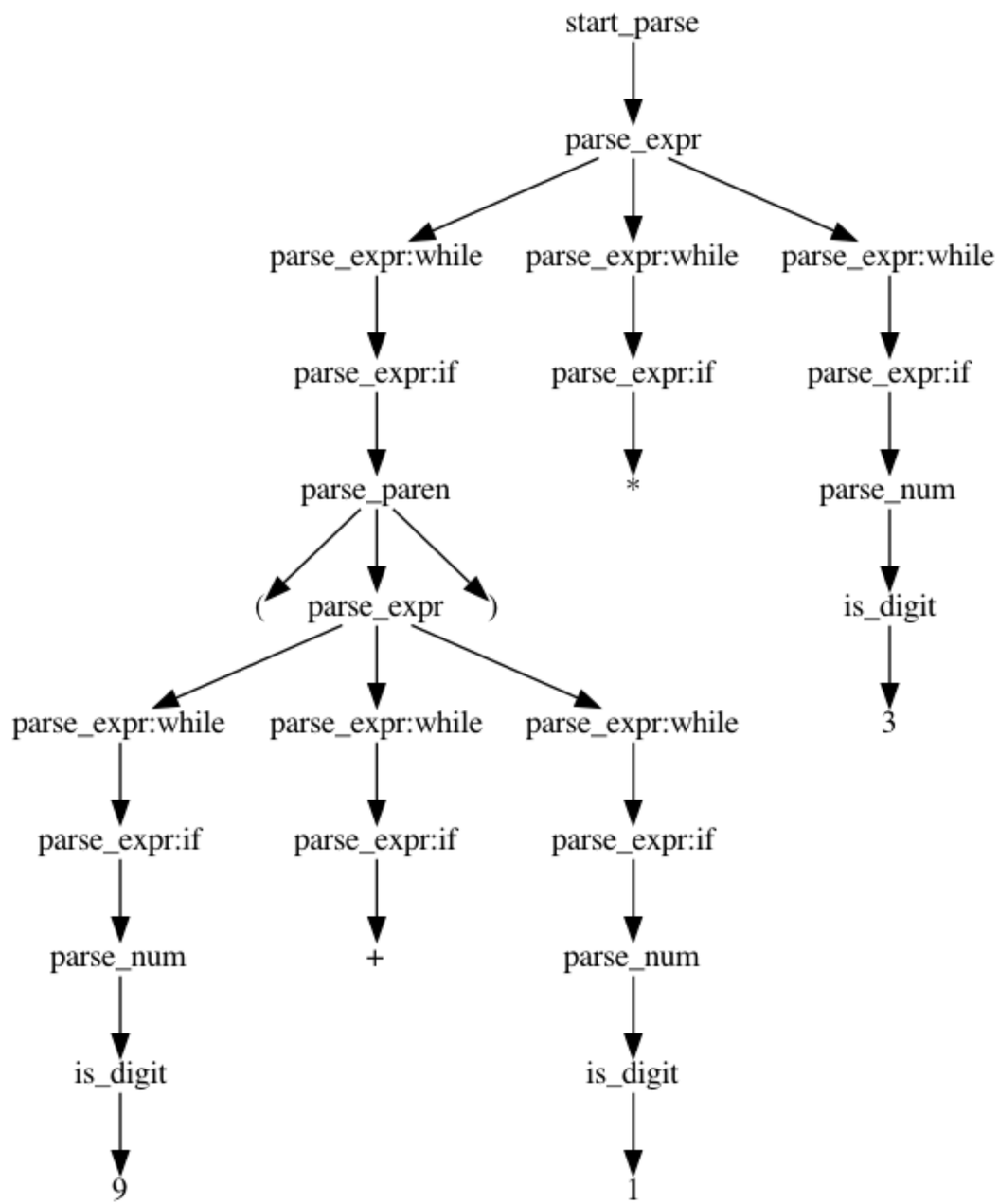
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

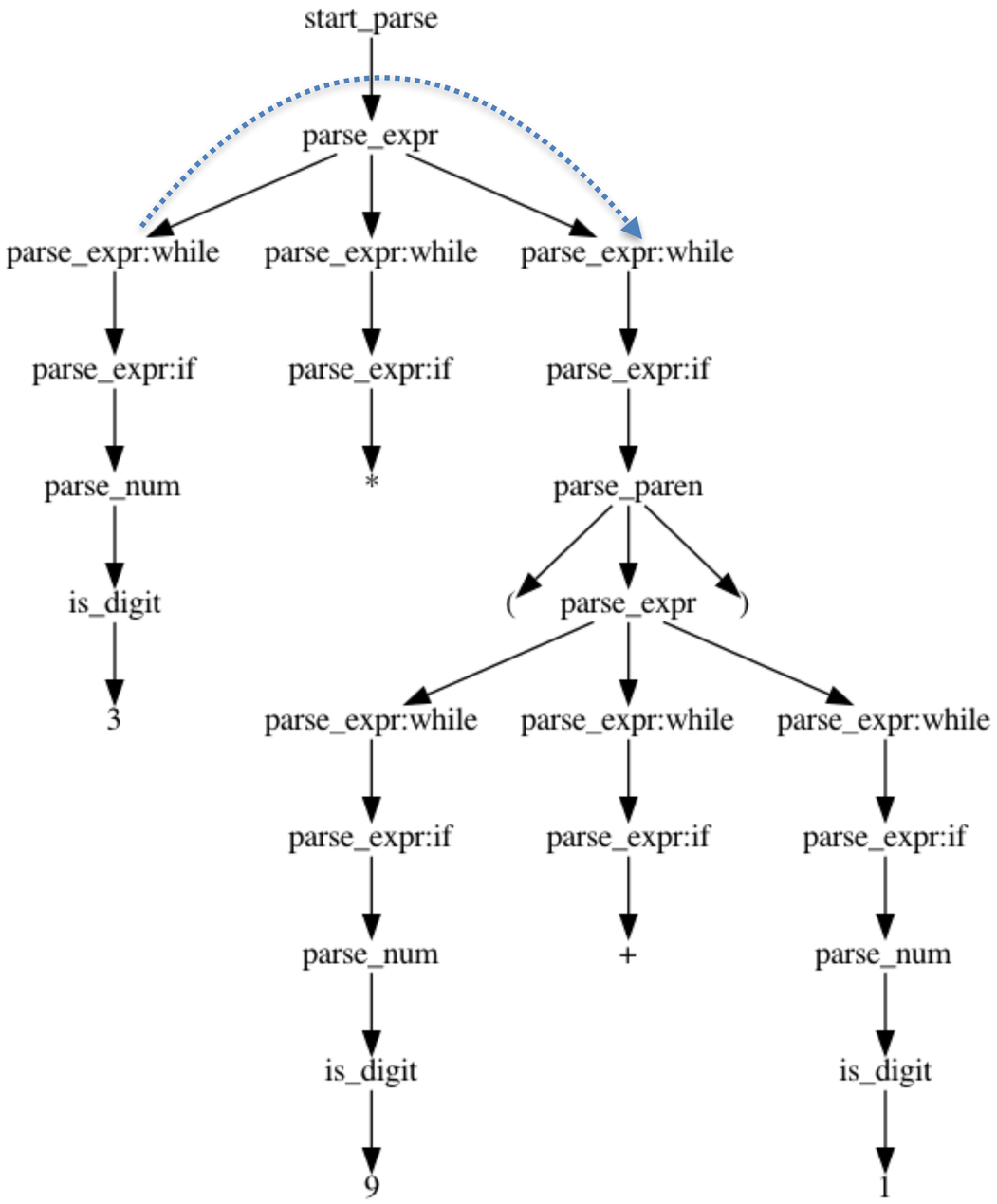
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

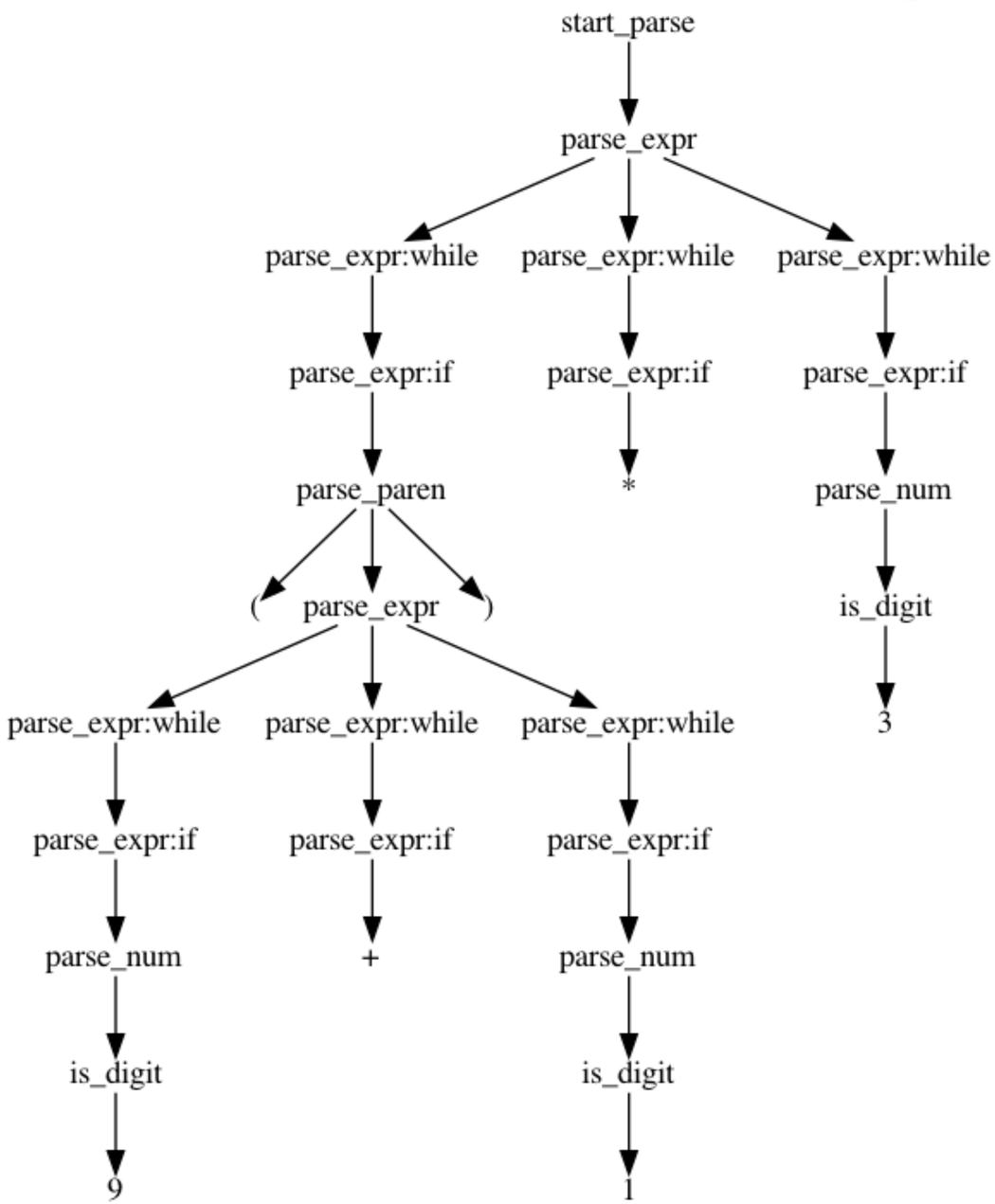
(9 + 1) * 3


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_paren
*
parse_num
(
parse_expr
)
is_digit
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

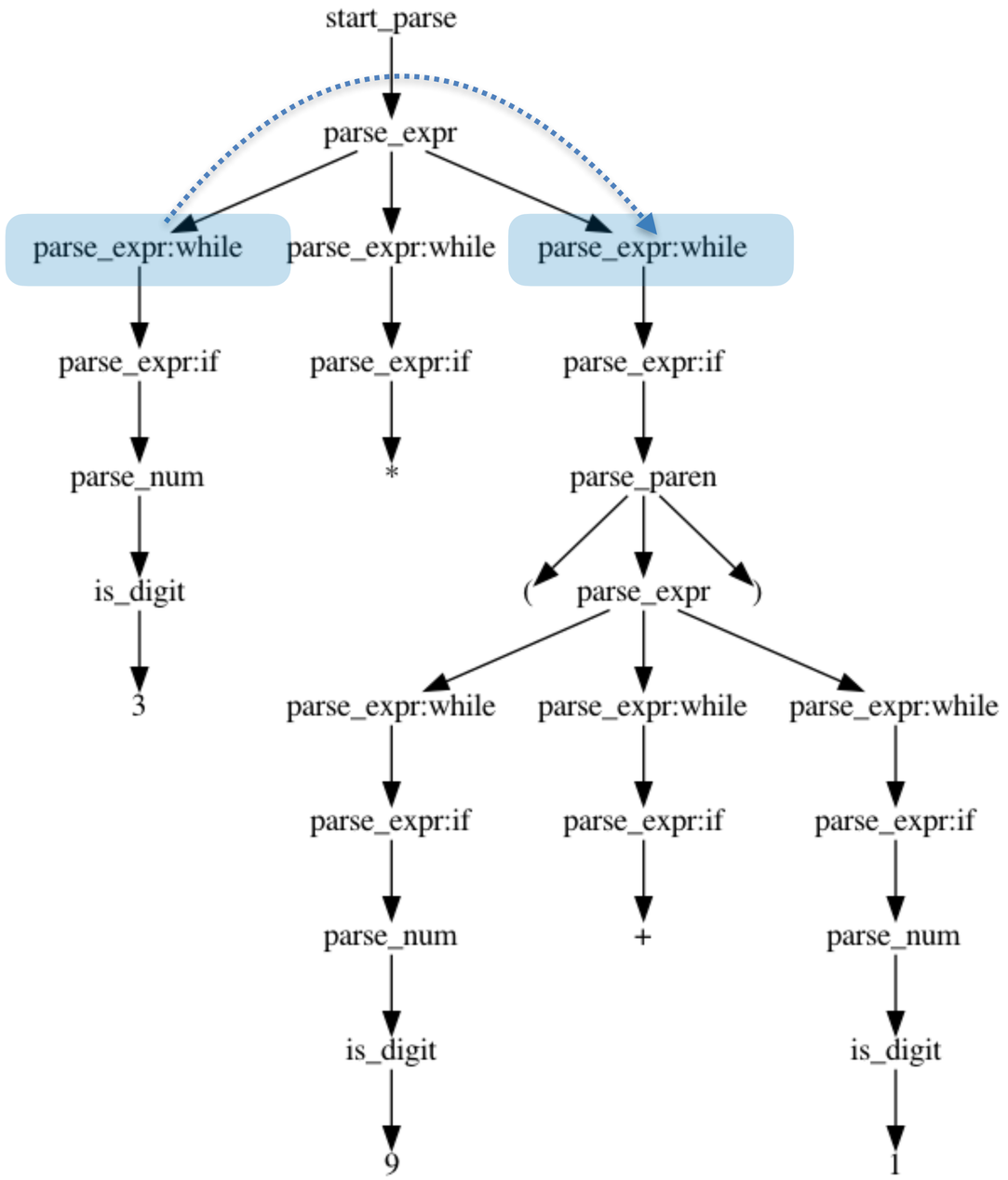
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

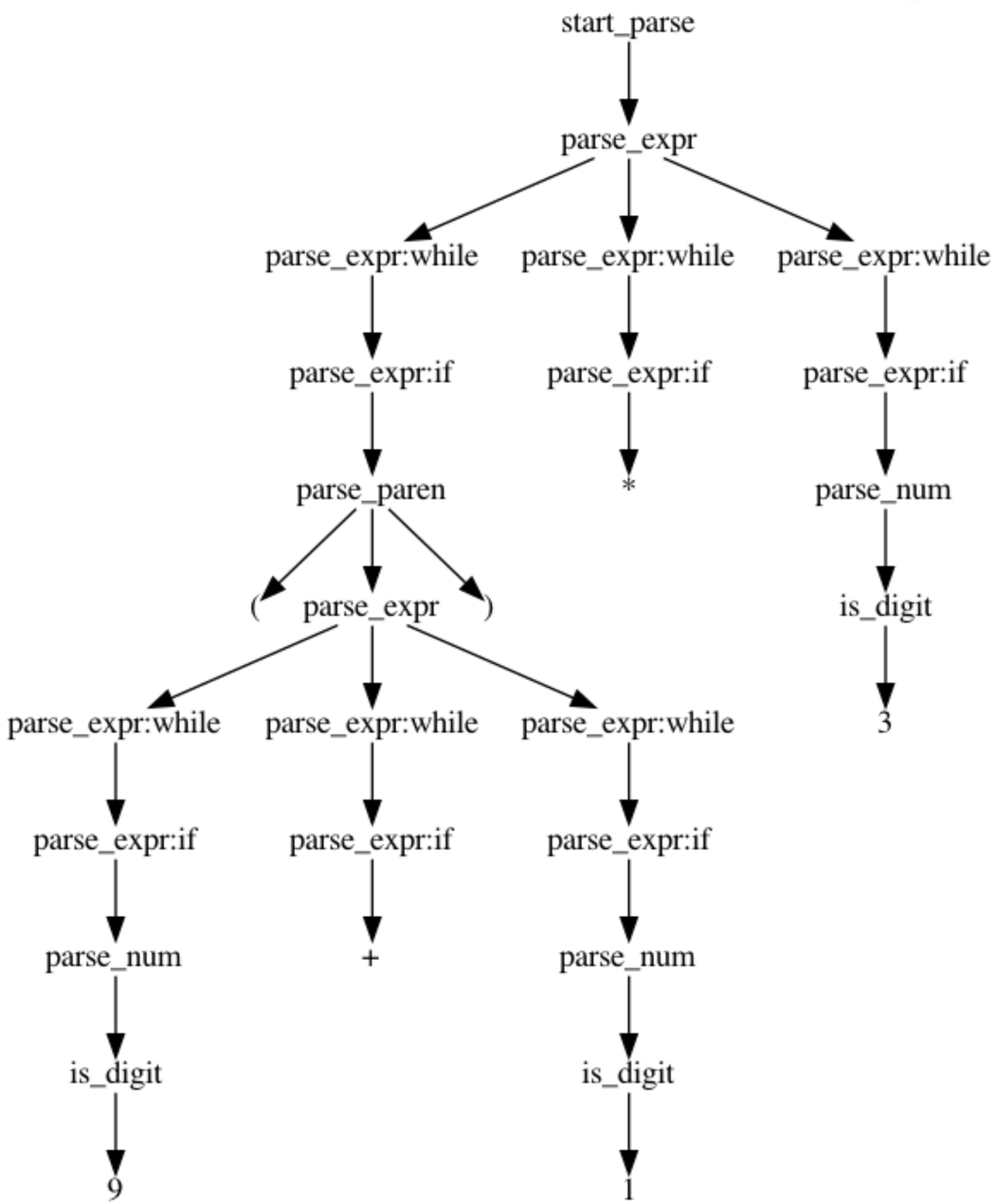
(9 + 1) * 3


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_paren
*
parse_num
(
parse_expr
)
is_digit
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

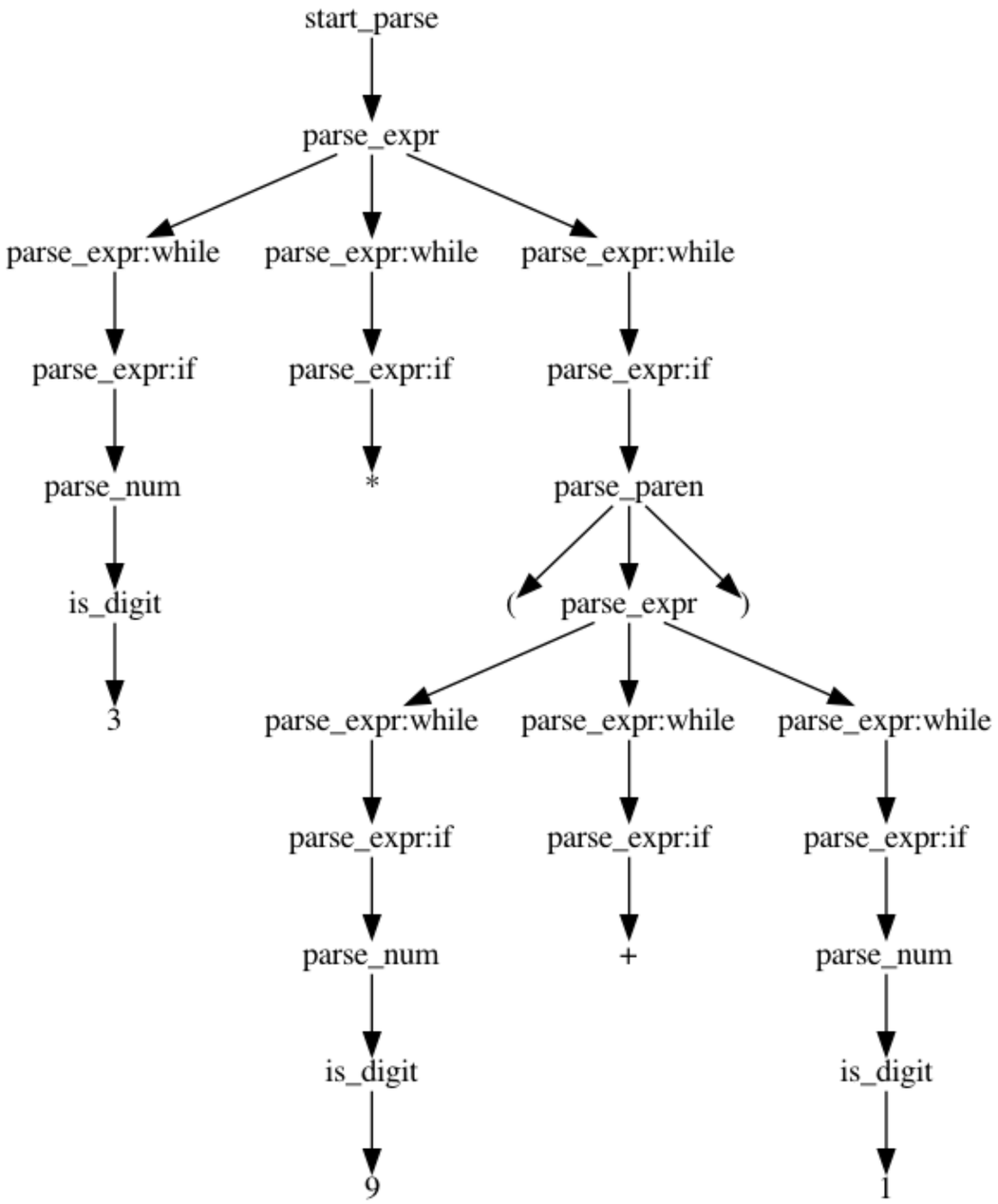
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

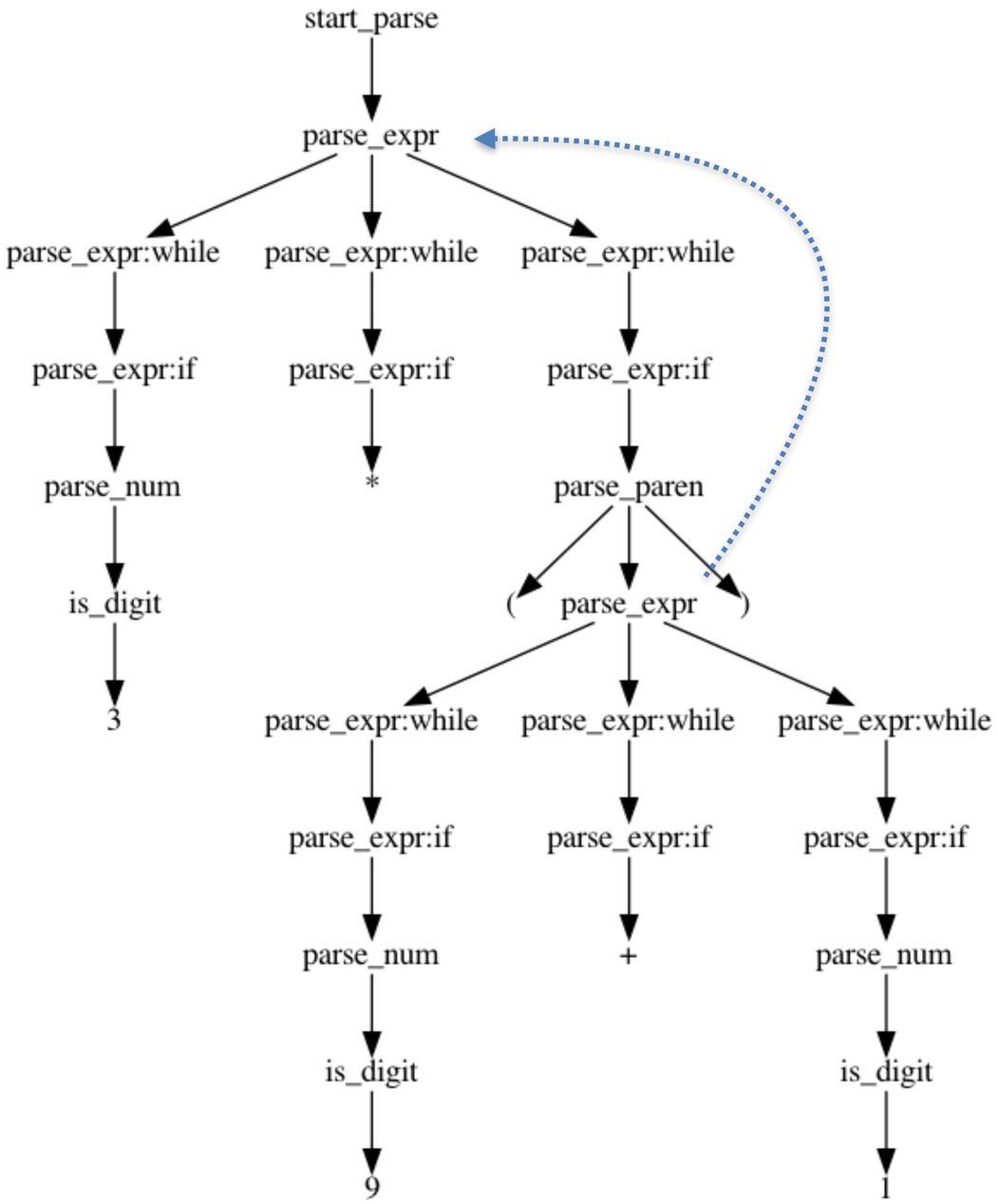
(9 + 1) * 3


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_paren
*
parse_num
(
parse_expr
)
is_digit
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

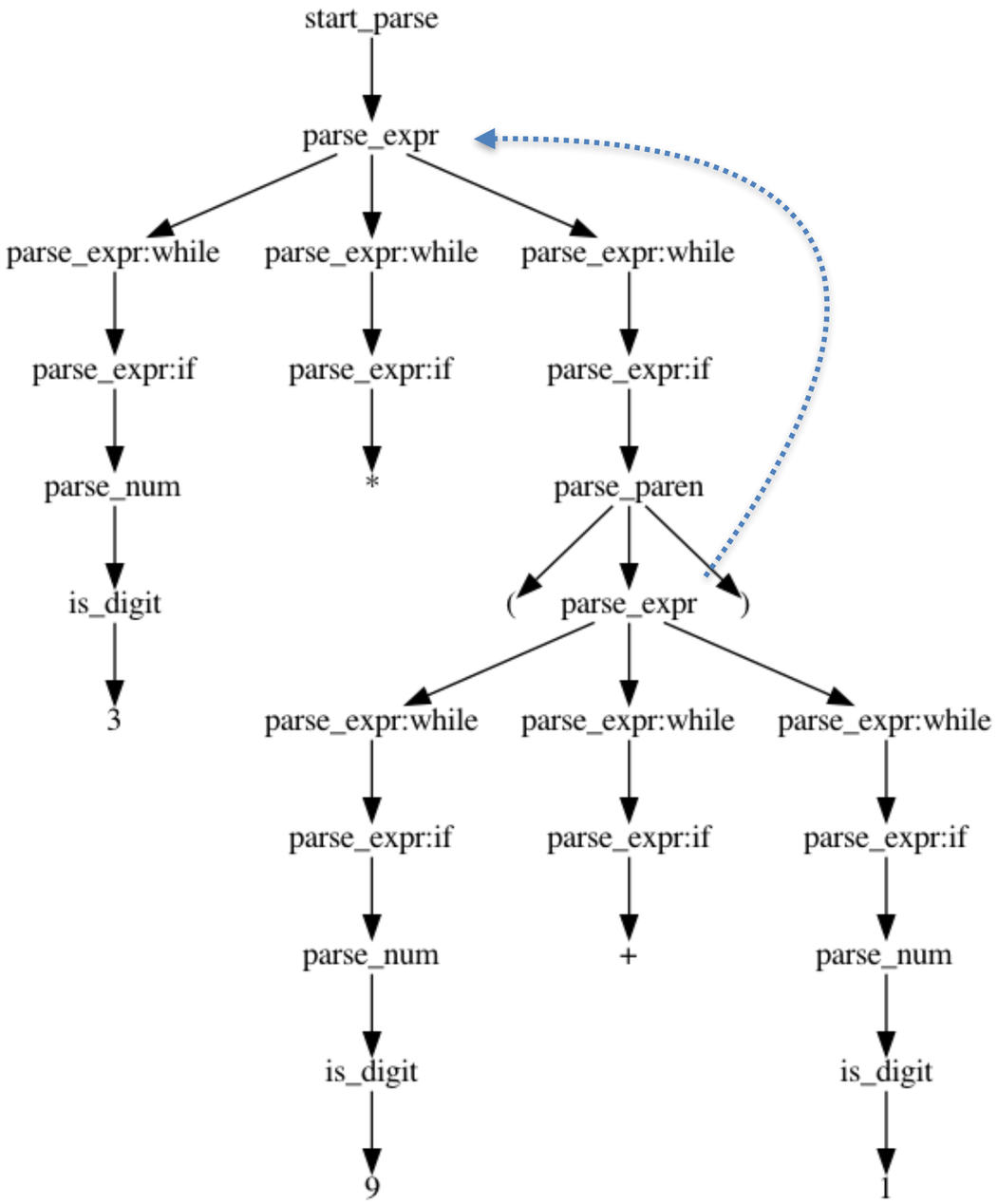
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

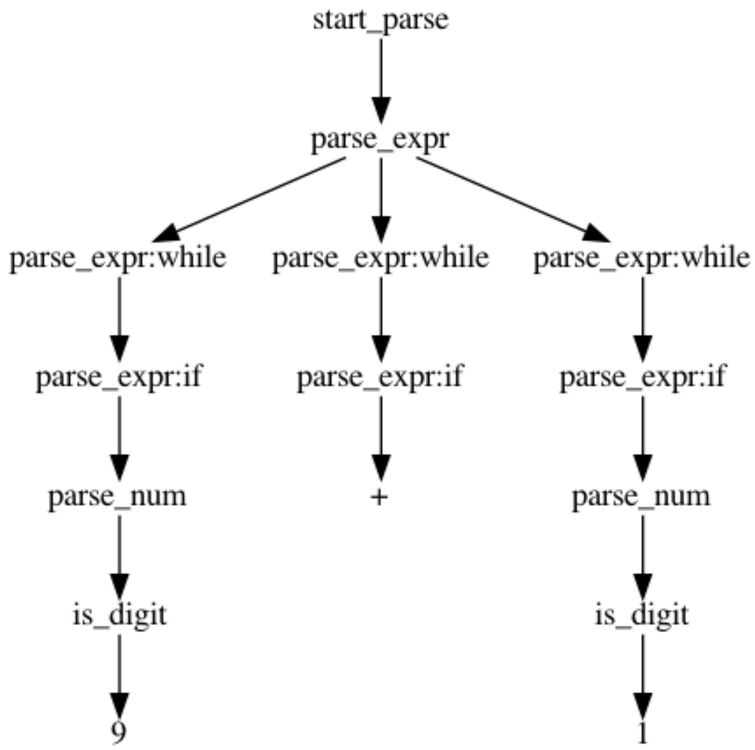
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

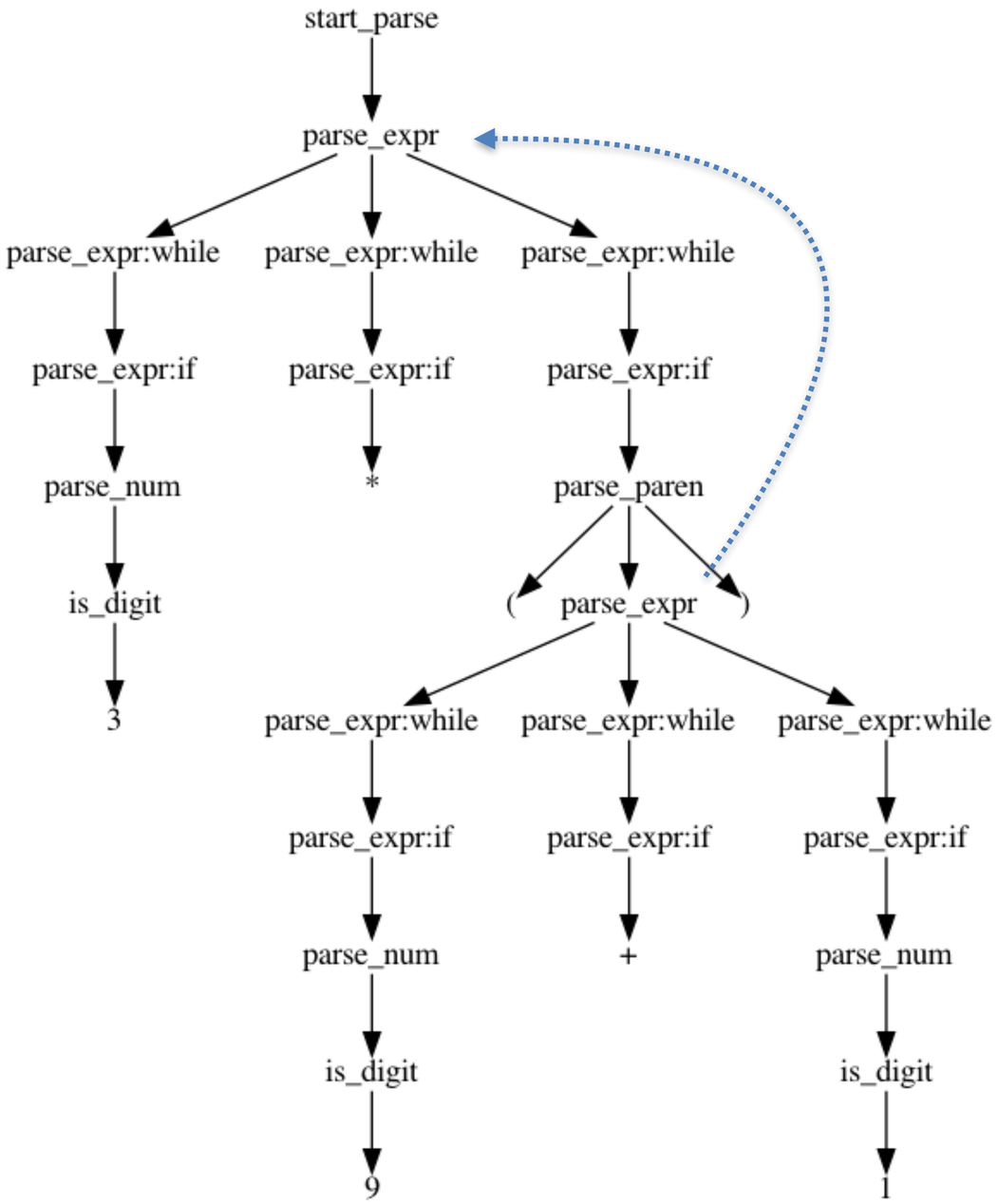
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

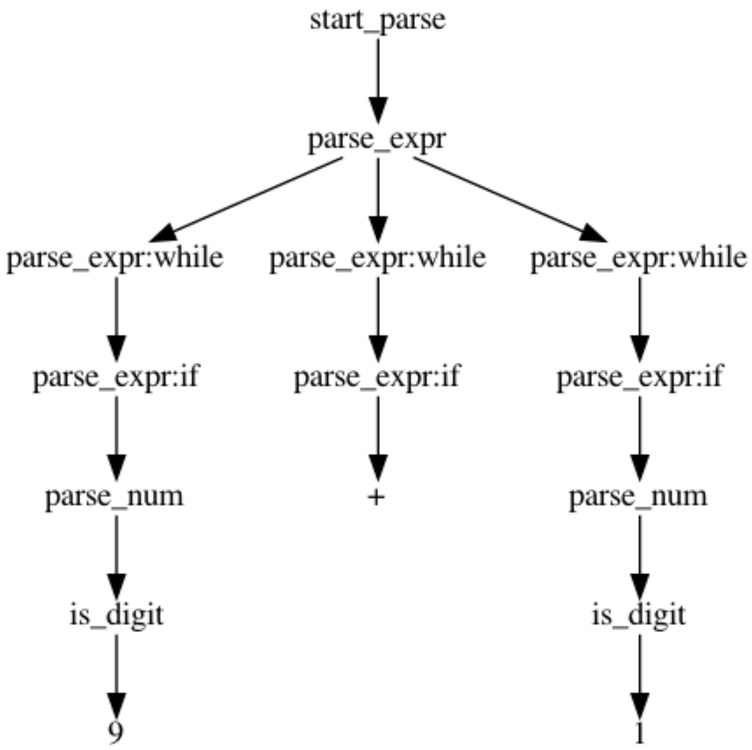
9 + 1


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

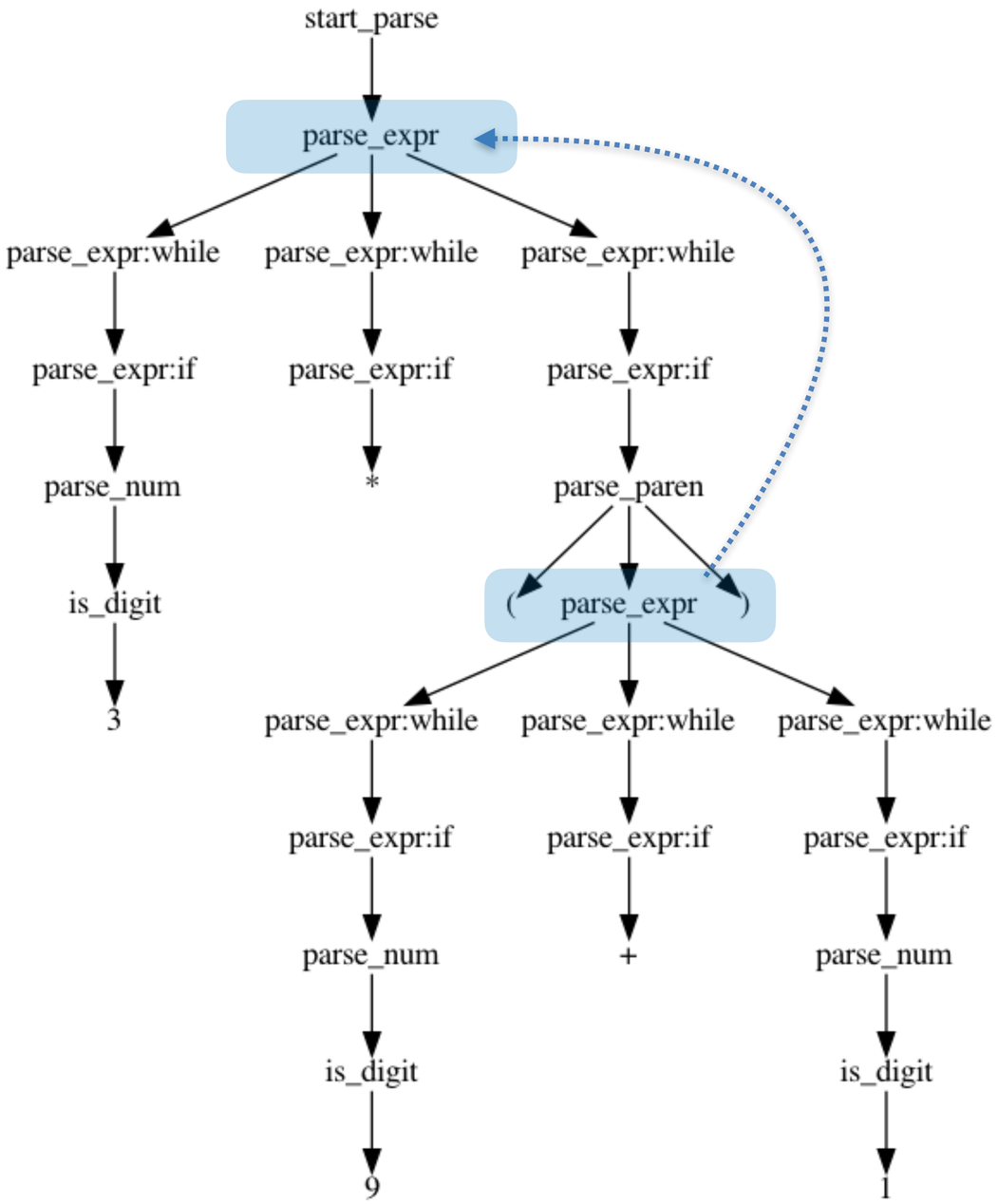
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

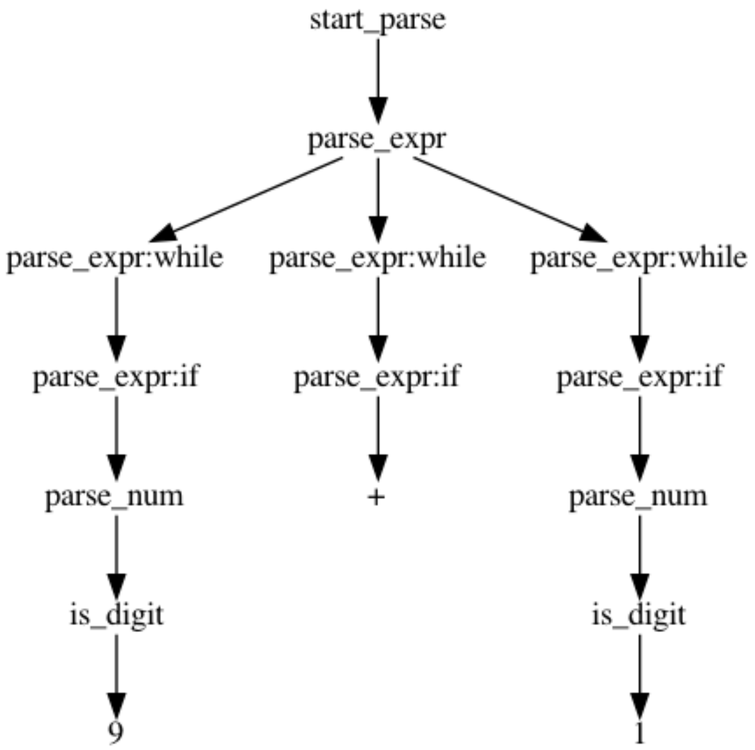
9 + 1


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

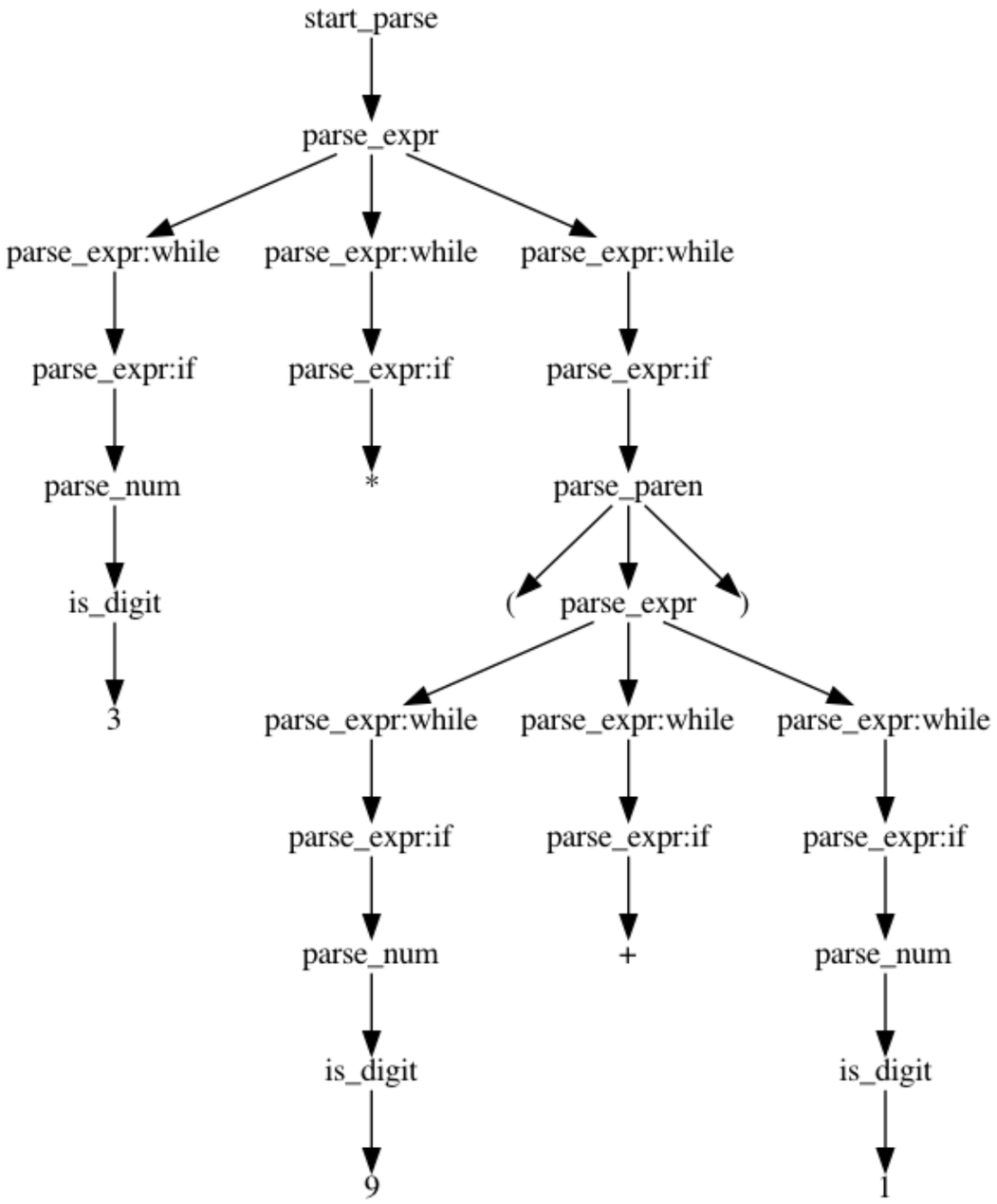
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

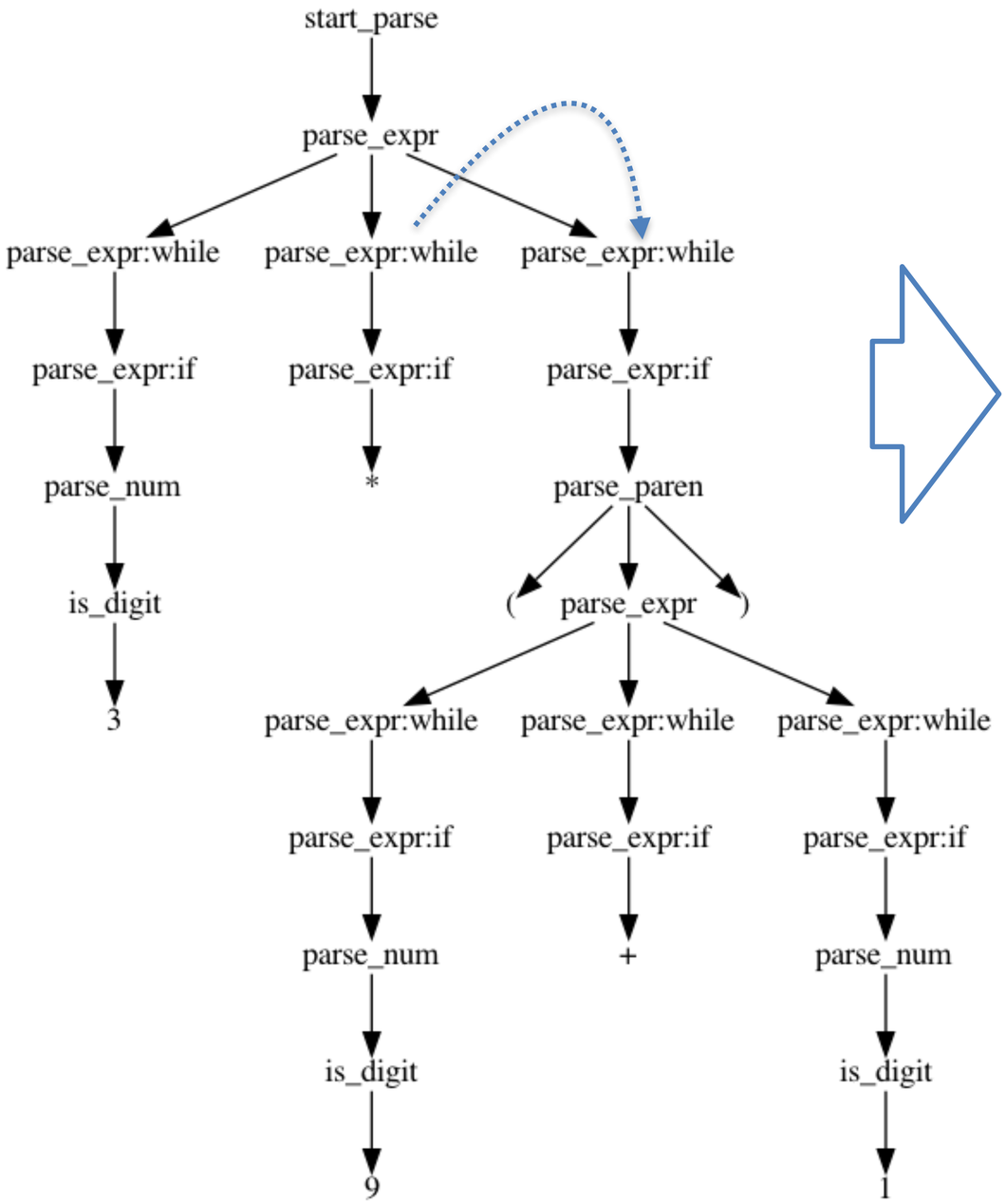
9 + 1


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

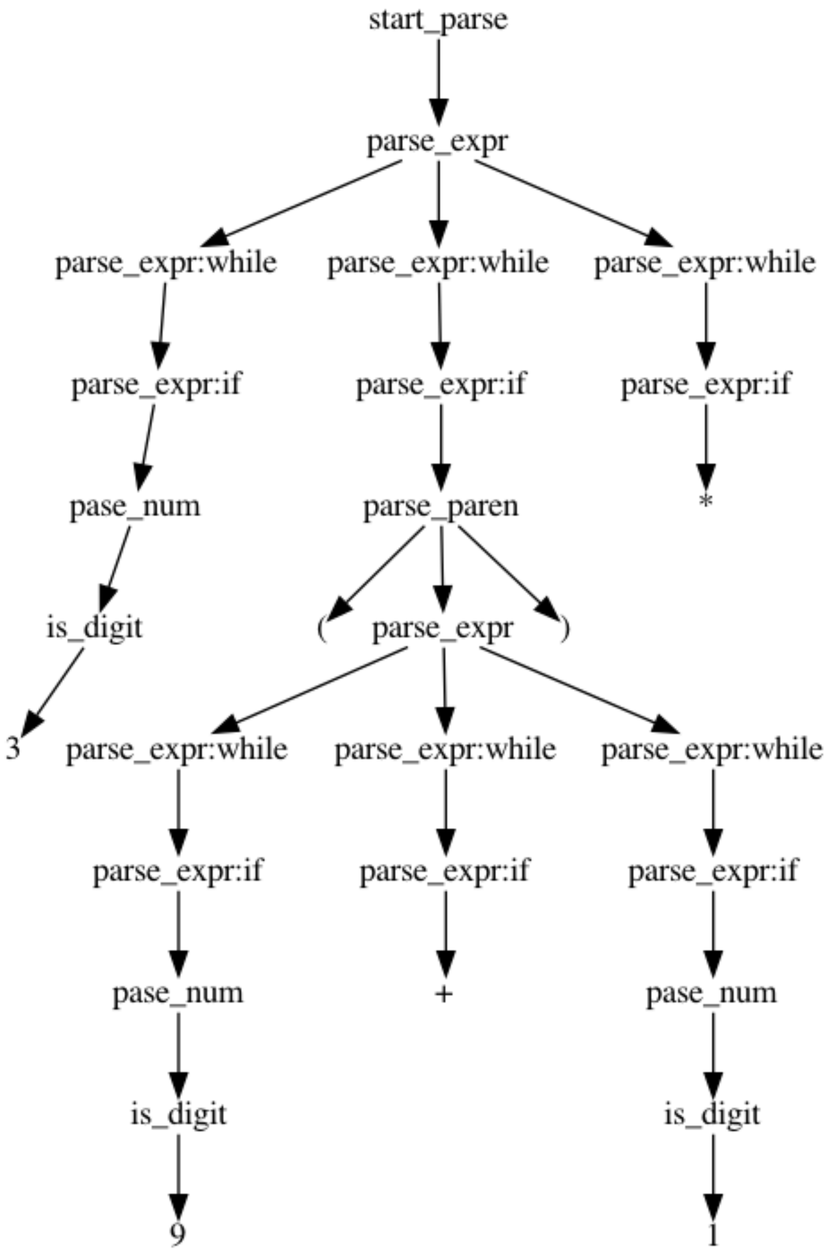
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

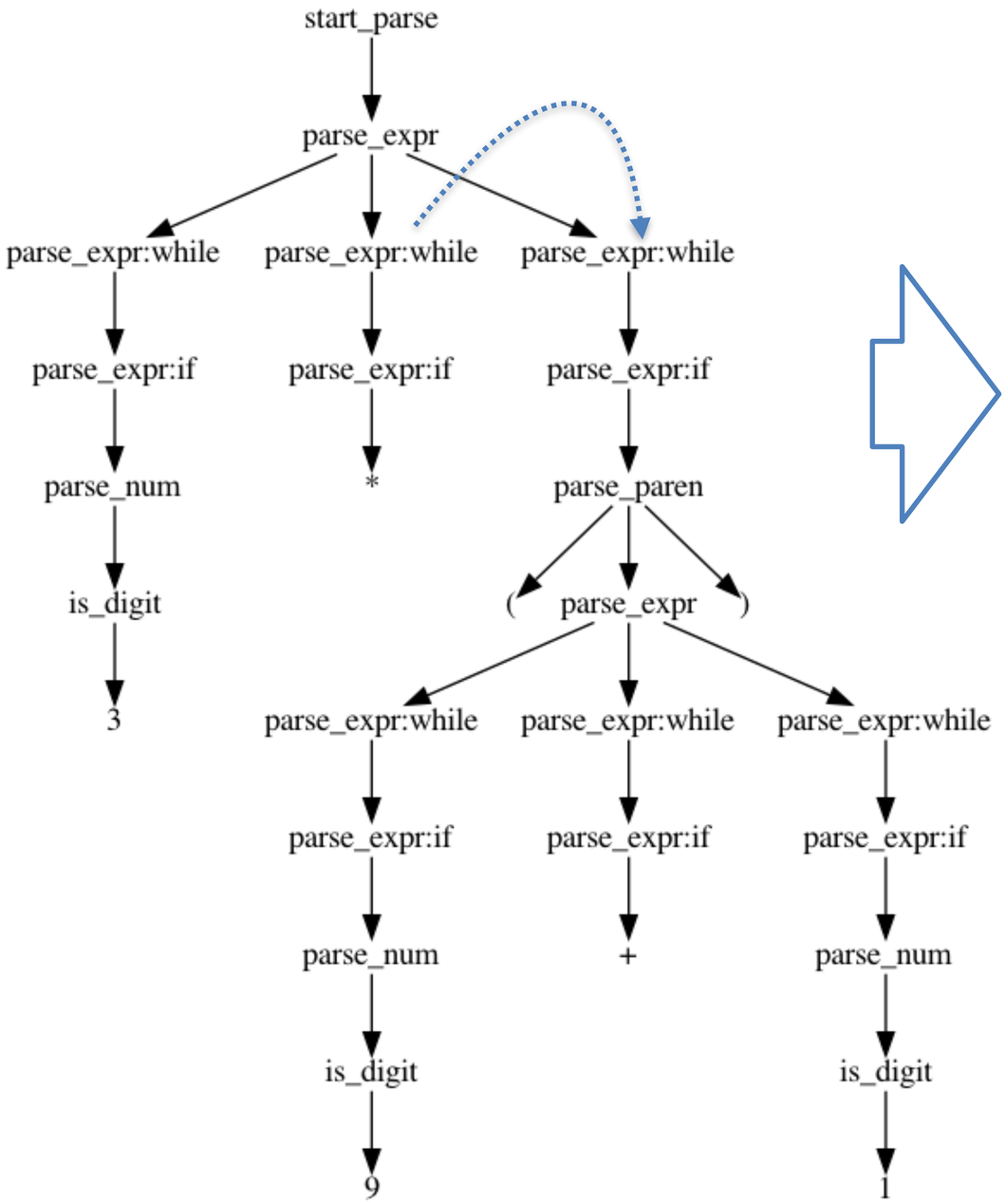
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

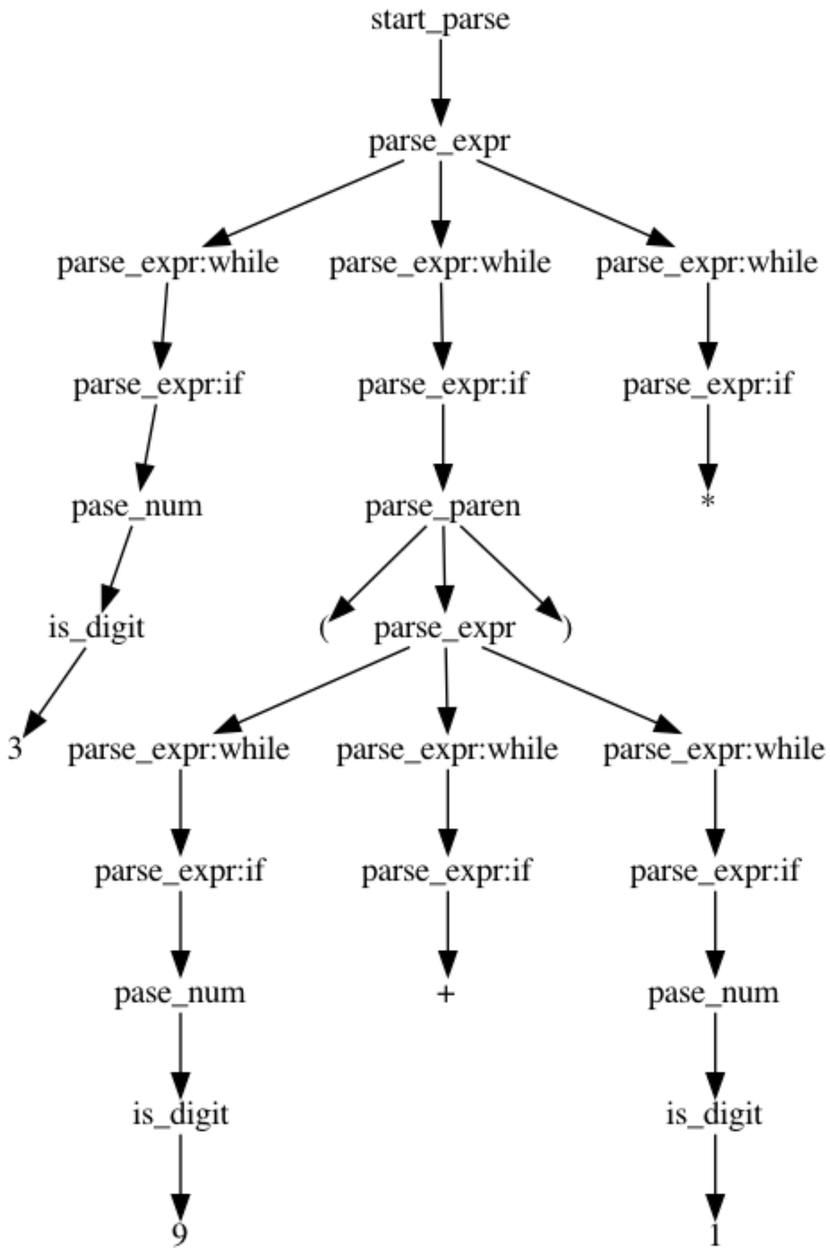
3 (9 + 1) *


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
pase_num
parse_paren
*
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
pase_num
+
pase_num
is_digit
is_digit
9
1

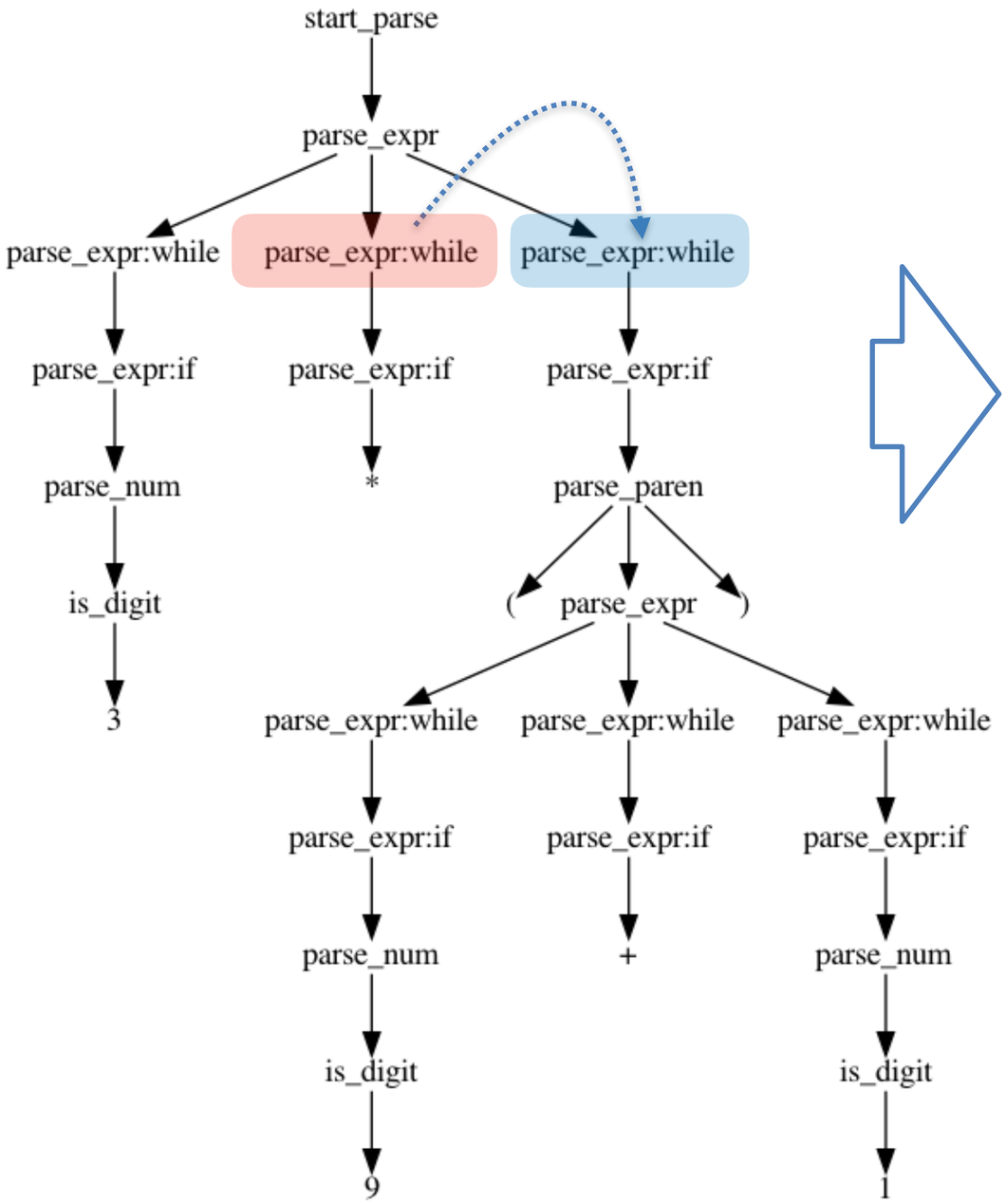
3 * (9 + 1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

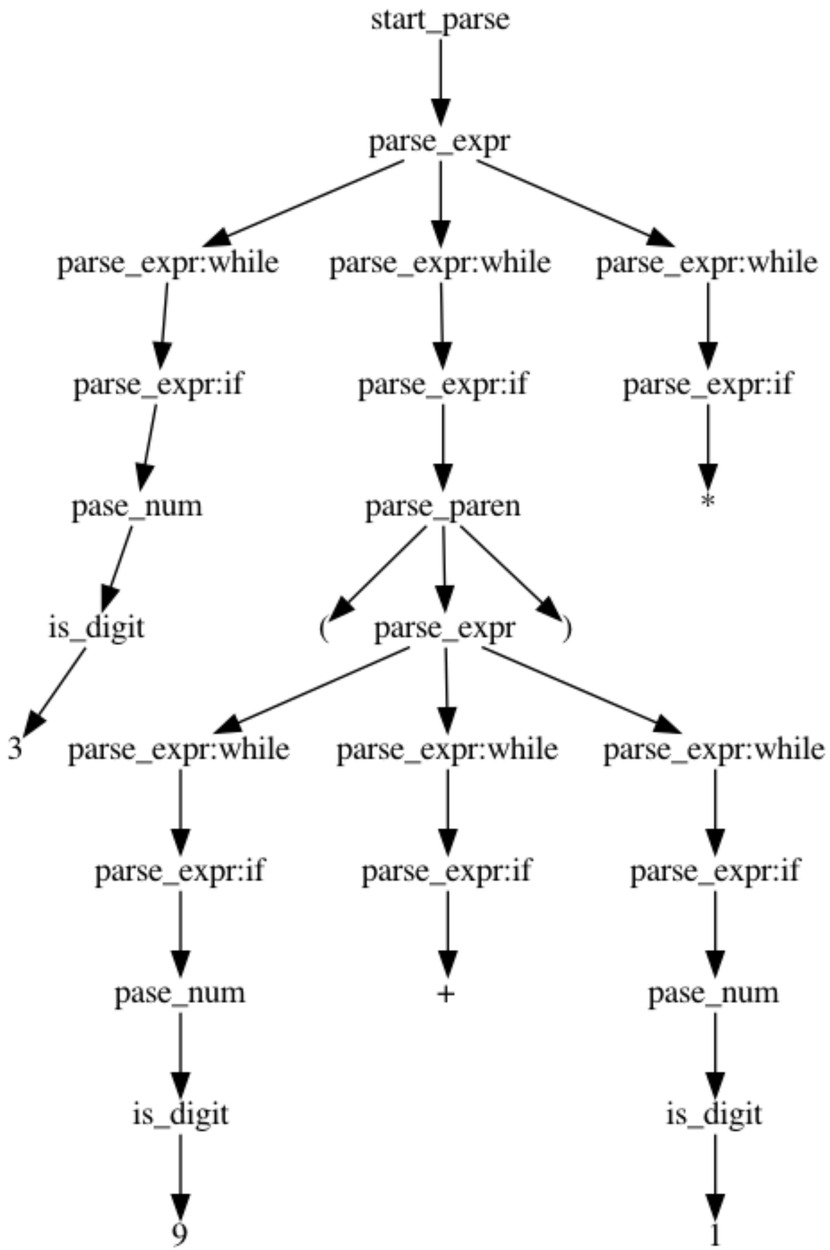
3 (9 + 1) *

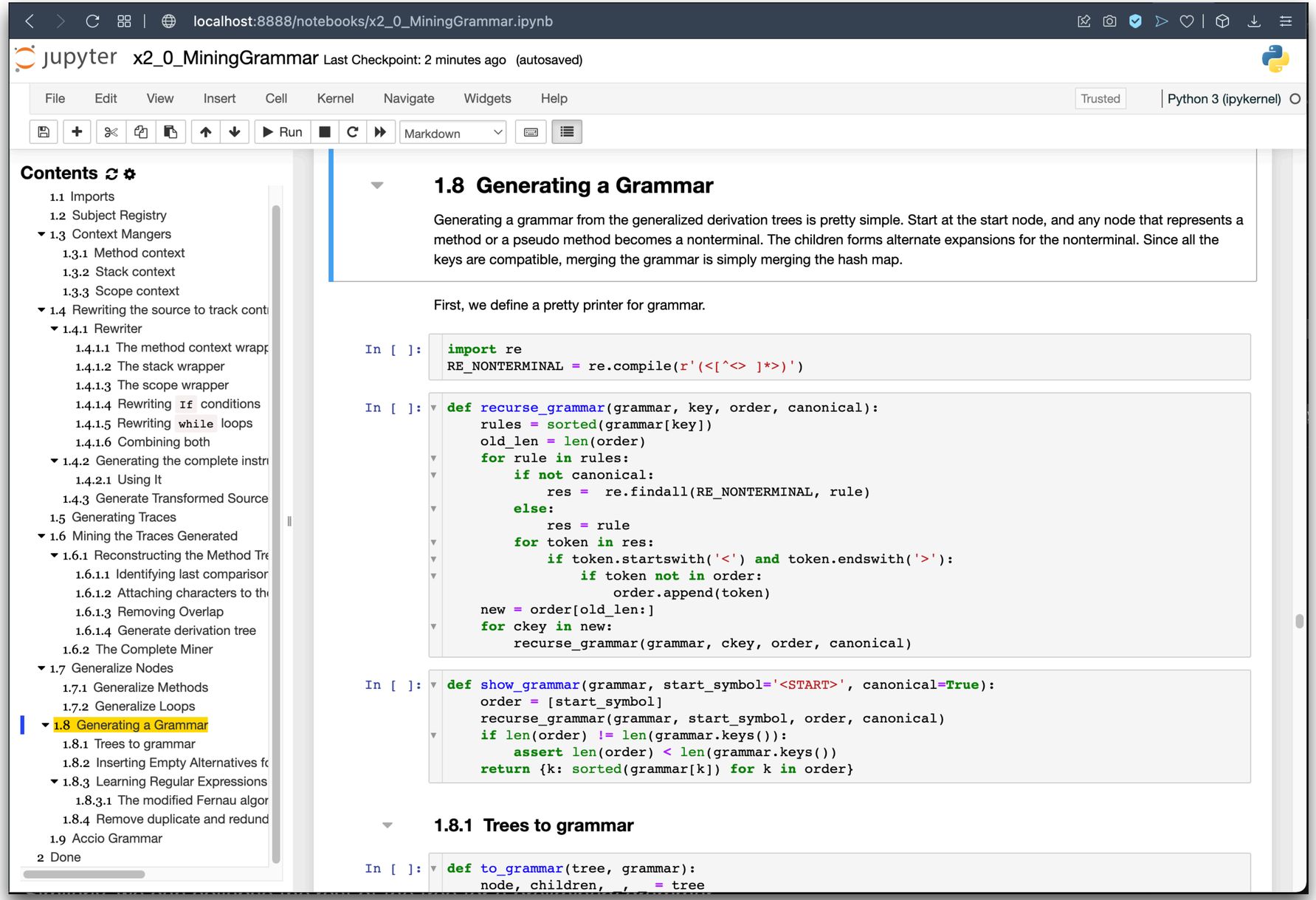

start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
pase_num
parse_paren
*
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
pase_num
+
pase_num
is_digit
is_digit
9
1

3 * (9 + 1)

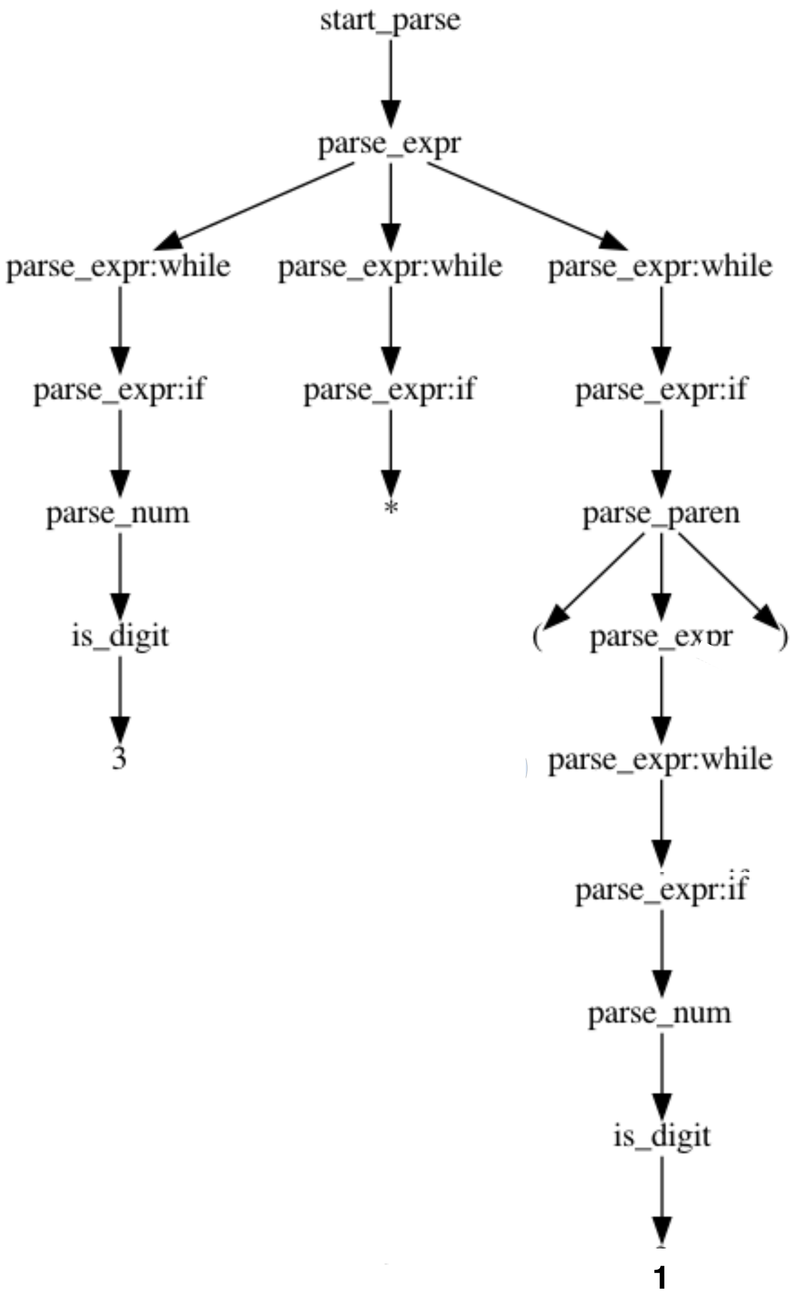

start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
+
parse_num
is_digit
is_digit
9
1

3 (9 + 1) *

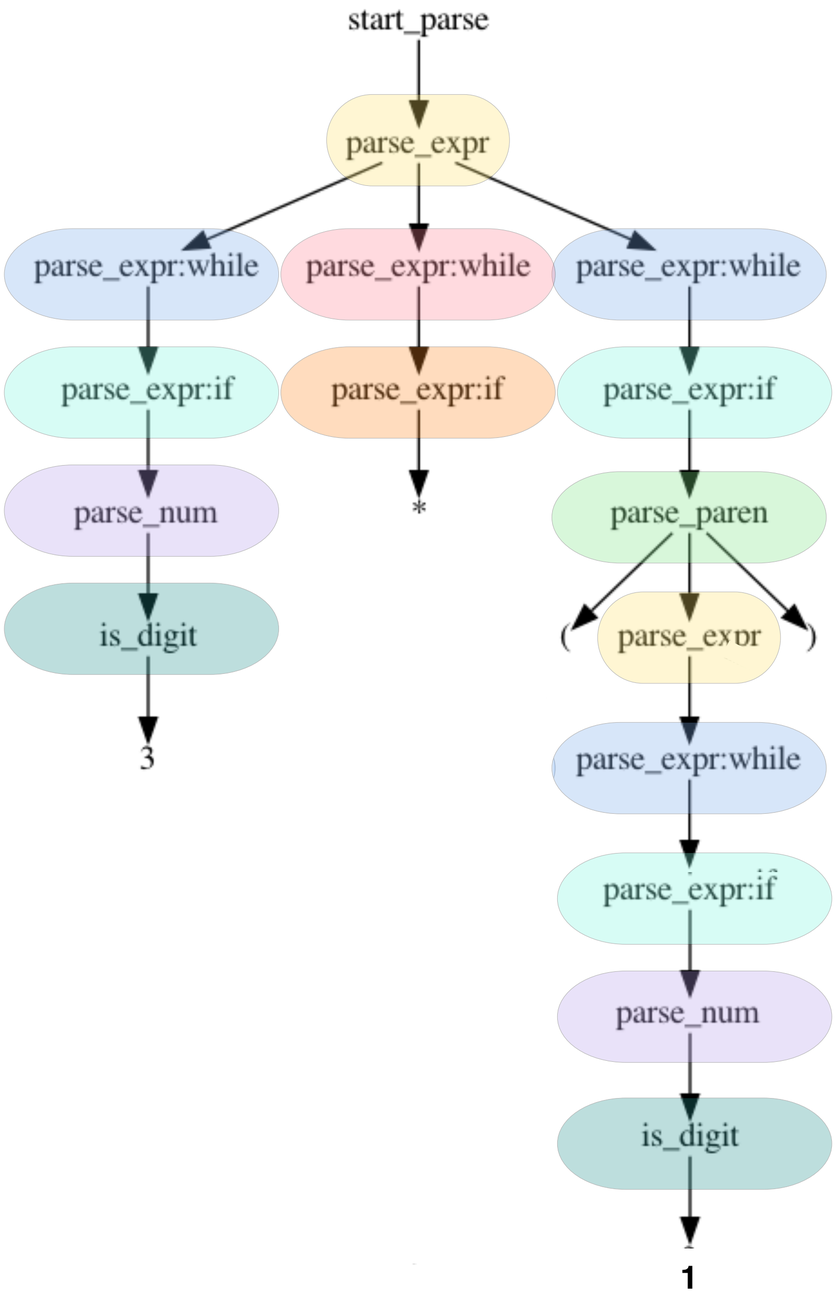

start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
pase_num
parse_paren
*
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
pase_num
+
pase_num
is_digit
is_digit
9
1


localhost:8888/notebooks/x2_0_MiningGrammar.ipynb
jupyter x2_0_MiningGrammar Last Checkpoint: 2 minutes ago (autosaved)
File Edit View Insert Cell Kernel Navigate Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
Contents
1.1 Imports
1.2 Subject Registry
1.3 Context Mangers
1.3.1 Method context
1.3.2 Stack context
1.3.3 Scope context
1.4 Rewriting the source to track conti
1.4.1 Rewriter
1.4.1.1 The method context wrapp
1.4.1.2 The stack wrapper
1.4.1.3 The scope wrapper
1.4.1.4 Rewriting If conditions
1.4.1.5 Rewriting while loops
1.4.1.6 Combining both
1.4.2 Generating the complete instru
1.4.2.1 Using It
1.4.3 Generate Transformed Source
1.5 Generating Traces
1.6 Mining the Traces Generated
1.6.1 Reconstructing the Method Tre
1.6.1.1 Identifying last comparisor
1.6.1.2 Attaching characters to the
1.6.1.3 Removing Overlap
1.6.1.4 Generate derivation tree
1.6.2 The Complete Miner
1.7 Generalize Nodes
1.7.1 Generalize Methods
1.7.2 Generalize Loops
1.8 Generating a Grammar
1.8.1 Trees to grammar
1.8.2 Inserting Empty Alternatives fo
1.8.3 Learning Regular Expressions
1.8.3.1 The modified Fernau algor
1.8.4 Remove duplicate and redund
1.9 Accio Grammar
2 Done
1.8 Generating a Grammar
Generating a grammar from the generalized derivation trees is pretty simple. Start at the start node, and any node that represents a method or a pseudo method becomes a nonterminal. The children forms alternate expansions for the nonterminal. Since all the keys are compatible, merging the grammar is simply merging the hash map.
First, we define a pretty printer for grammar.
In [ ]:
import re
RE_NONTERMINAL = re.compile(r'(<[^<>]*>)')
In [ ]:
def recurse_grammar(grammar, key, order, canonical):
rules = sorted(grammar[key])
old_len = len(order)
for rule in rules:
if not canonical:
res = re.findall(RE_NONTERMINAL, rule)
else:
res = rule
for token in res:
if token.startswith('<') and token.endswith('>'):
if token not in order:
order.append(token)
new = order[old_len:]
for ckey in new:
recurse_grammar(grammar, ckey, order, canonical)
In [ ]:
def show_grammar(grammar, start_symbol='<START>', canonical=True):
order = [start_symbol]
recurse_grammar(grammar, start_symbol, order, canonical)
if len(order) != len(grammar.keys()):
assert len(order) < len(grammar.keys())
return {k: sorted(grammar[k]) for k in order}
1.8.1 Trees to grammar
In [ ]:
def to_grammar(tree, grammar):
node, children, _ = tree

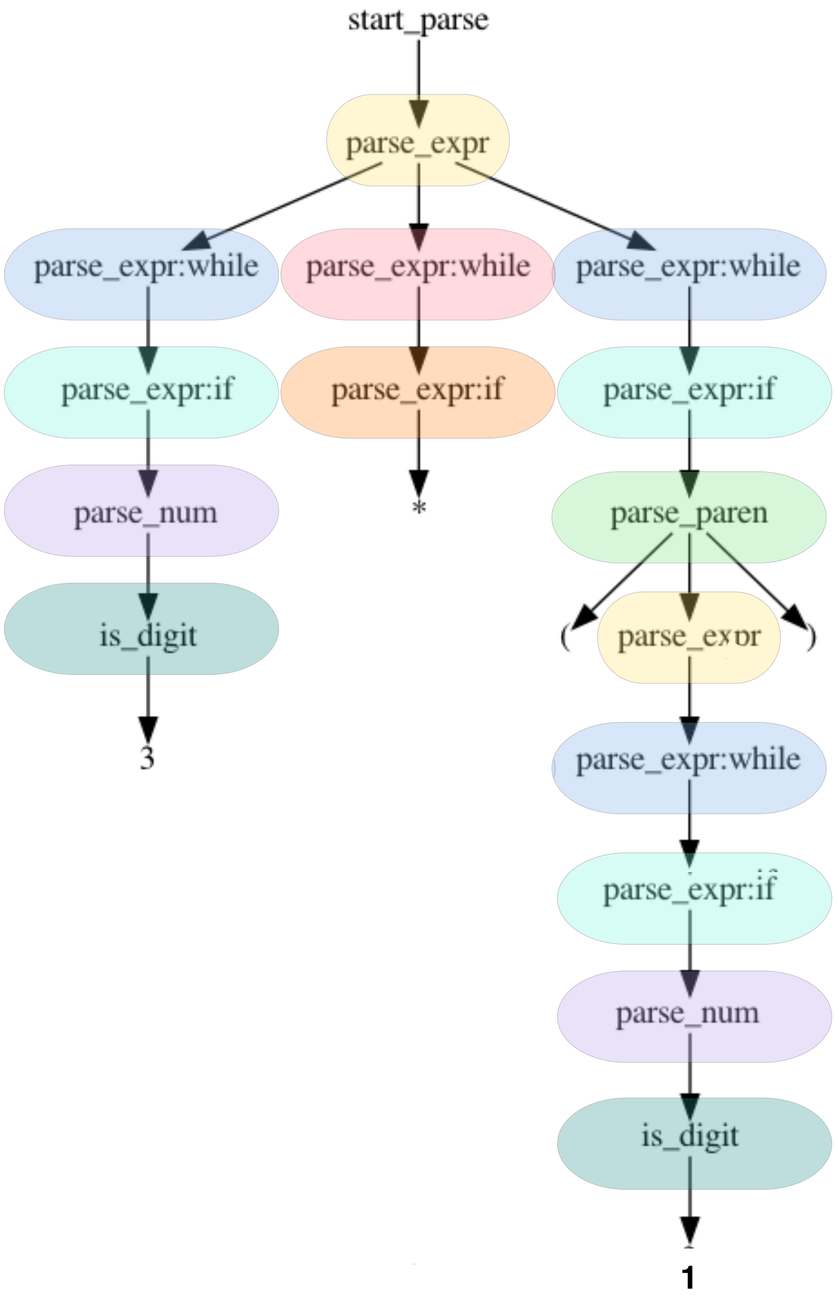
3*(1)


start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:if
parse_num
is_digit
1

3*(1)

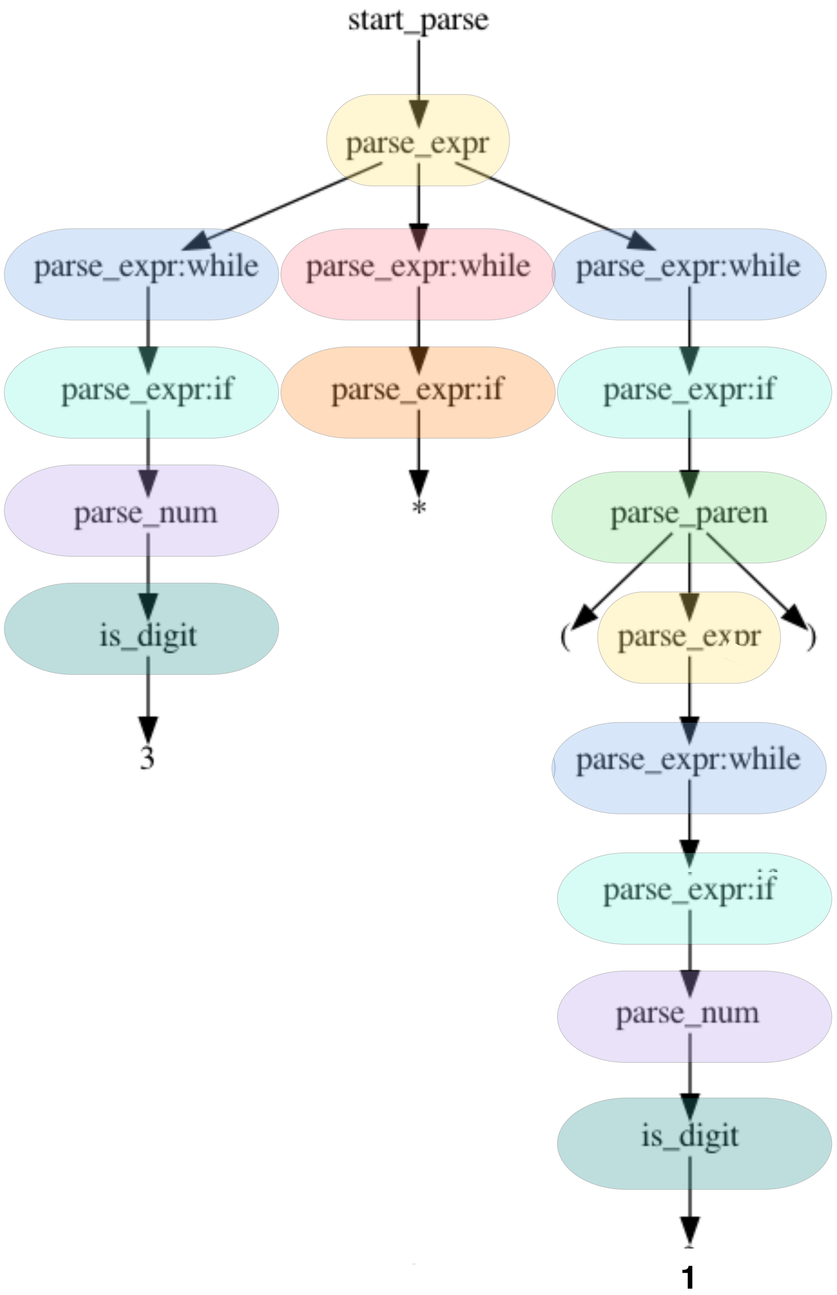

start_parse
parse_expr
parse_expr:while
parse_expr:while
parse_expr:while
parse_expr:if
parse_expr:if
parse_expr:if
parse_num
*
parse_paren
is_digit
(
parse_expr
)
3
parse_expr:while
parse_expr:if
parse_num
is_digit
1

3*(1)



<parse_expr> := <while 1:1> <while 1:0> <while 1:1>

3*(1)



<parse_expr> := <while 1:1> <while 1:0> <while 1:1>
<while 1:1>

```
<parse_expr> := <while 1:1> <while 1:0> <while 1:1>
              |  <while 1:1> <while 1:0> <while 1:1> <while 1:0> <while 1:1> <while 1:0> <while 1:1>
              |  <while 1:1> <while 1:0> <while 1:1> <while 1:0> <while 1:1>
              |  <while 1:1>
<while 1:1>  := <if 1:1>
<if 1:1>     := <parse_num>
              |  <parse_paren>
<parse_num>  := <is_digit>
<is_digit>   := '3' | '1'

<parse_paren>:= '(' <parse_expr> ')'

<while 1:0>  := <if 1:0>
<if 1:0>     := '*'
```

```
<parse_expr> := <while 1:1> <while 1:0> <while 1:1>
              |  <while 1:1> <while 1:0> <while 1:1> <while 1:0> <while 1:1> <while 1:0> <while 1:1>
              |  <while 1:1> <while 1:0> <while 1:1> <while 1:0> <while 1:1>
              |  <while 1:1>
<while 1:1>  := <if 1:1>
<if 1:1>     := <parse_num>
              |  <parse_paren>
<parse_num>  := <is_digit>
<is_digit>   := '3' | '1'

<parse_paren>:= '(' <parse_expr> ')'

<while 1:0>  := <if 1:0>
<if 1:0>     := '*'
```

```
<parse_expr> := <while_s>

<while_s>    := <while_1:1> <while_1:0> <while_s>
              |  <while_1:1>
```

```
def is_digit(i): return i in '0123456789'
def parse_num(s,i):
    n = ''
    while s[i:] and is_digit(s[i]):
        n += s[i]
        i = i +1
    return i,n

def parse_paren(s, i):
    assert s[i] == '('
    i, v = parse_expr(s, i+1)
    if s[i:] == '': raise Ex(s, i)
    assert s[i] == ')'
    return i+1, v

def parse_expr(s, i = 0):
    expr, is_op = [], True
    while s[i:]:
        c = s[i]
        if isdigit(c):
            if not is_op: raise Ex(s,i)
            i,num = parse_num(s,i)
            expr.append(num)
            is_op = False
        elif c in ['+', '-', '*', '/']:
            if is_op: raise Ex(s,i)
            expr.append(c)
            is_op, i = True, i + 1
        elif c == '(':
            if not is_op: raise Ex(s,i)
            i, cexpr = parse_paren(s, i)
            expr.append(cexpr)
            is_op = False
        elif c == ')': break
        else: raise Ex(s,i)
    if is_op: raise Ex(s,i)
    return i, expr
```

```
<START>             := <parse_expr.0-0-c>

<parse_expr.0-0-c>  := <parse_expr.0-1-s><parse_expr.0>
                    |  <parse_expr.0>

<parse_expr.0-1-s>  := <parse_expr.0><parse_expr.0-2>
                    |  <parse_expr.0><parse_expr.0-2><parse_expr.0-1-s>

<parse_expr.0>      := '(' <parse_expr.0-0-c> ')'
                    |  <parse_num.0-1-s>

<parse_expr.0-2>    := '*' | '+' | '-' | '/'

<parse_num.0-1-s>   := <is_digit.0-0-c>
                    |  <is_digit.0-0-c><parse_num.0-1-s>

<is_digit.0-0-c>    : [0-9]
```

calc.py **Recovered Arithmetic Grammar**

```
<START>             := <parse_expr.0-0-c>

<parse_expr.0-0-c> := <parse_expr.0-1-s><parse_expr.0>
                    |  <parse_expr.0>

<parse_expr.0-1-s> := <parse_expr.0><parse_expr.0-2>
                    |  <parse_expr.0><parse_expr.0-2><parse_expr.0-1-s>

<parse_expr.0>     := '(' <parse_expr.0-0-c> ')'
                    |  <parse_num.0-1-s>

<parse_expr.0-2>   := '*' | '+' | '-' | '/'

<parse_num.0-1-s>  := <is_digit.0-0-c>
                    |  <is_digit.0-0-c><parse_num.0-1-s>

<is_digit.0-0-c>   : [0-9]
```

```
<START>              := <parse_expr.0-0-c>

<parse_expr.0-0-c> := <parse_expr.0-1-s><parse_expr.0>
                     |  <parse_expr.0>

<parse_expr.0-1-s> := <parse_expr.0><parse_expr.0-2>
                     |  <parse_expr.0><parse_expr.0-2><parse_expr.0-1-s>

<parse_expr.0>     := '(' <parse_expr.0-0-c> ')'
                     |  <parse_num.0-1-s>

<parse_expr.0-2>   := '*' | '+' | '-' | '/'

<parse_num.0-1-s>  := <is_digit.0-0-c>
                     |  <is_digit.0-0-c><parse_num.0-1-s>

<is_digit.0-0-c>   : [0-9]
```

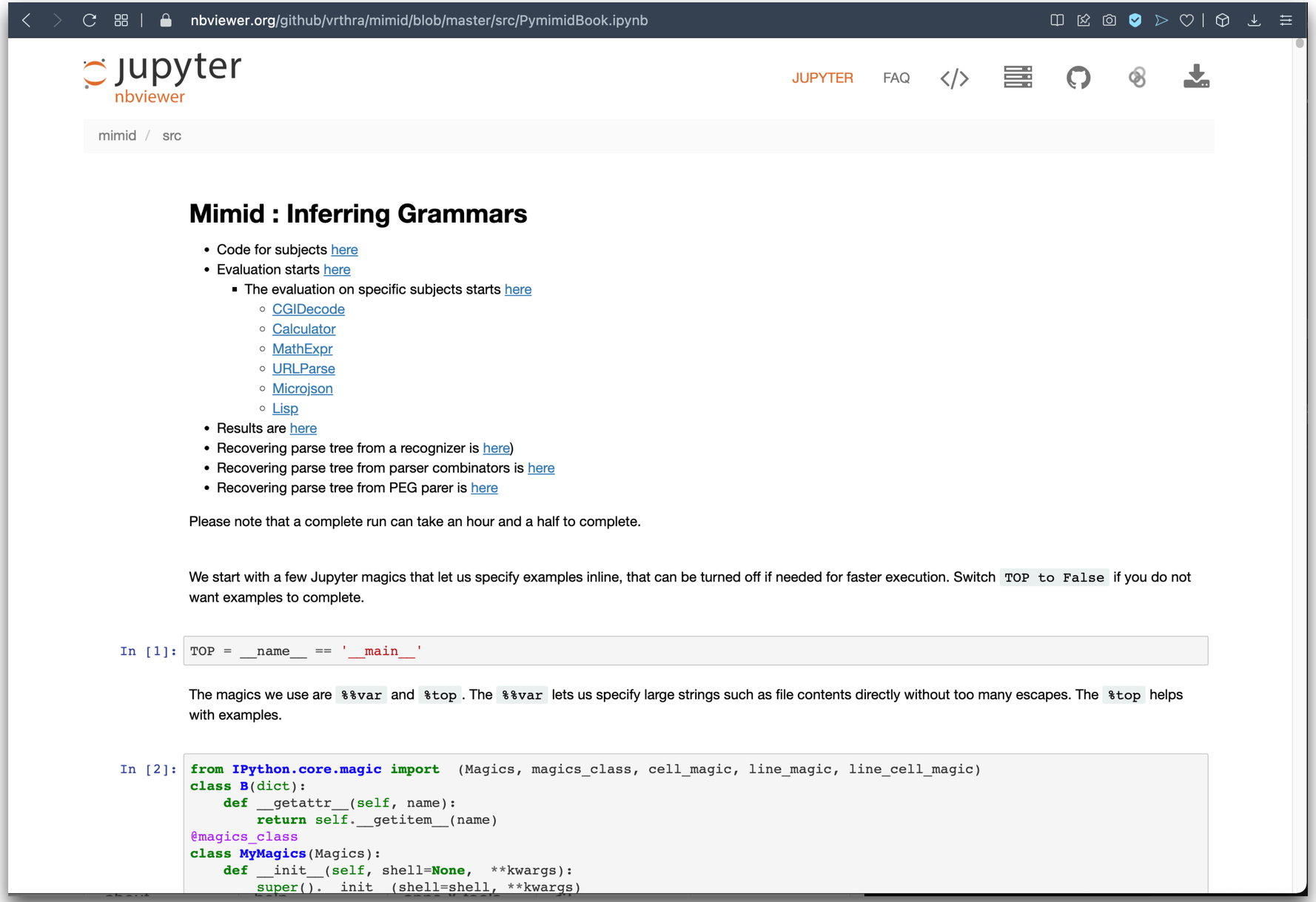
```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-
((-1 * +(8 - 5 - 6)) * (-(a-+(((+(4)
)))) - ++4) / +(-+---((5.6 - --(3 *
-1.8 * +(6 * +-(((-(-6) * ---+6)) /
+--(+-+-7 * (-0 * (+(((((2)) + 8 - 3
 - ++9.0 + ---(--+7 / (1 / +++6.37)
+ (1) / 482) / +++-+0)))) + 8.2 - 27
 - -9 / +((+9 * --2 + --+-+-((-1 * +
(8 - 5 - 6)) * (-(a-+(((+(4))))) - +
+4) / +(-+---((5.6 - --(3 * -1.8 * +
(6 * +-(((-(-6) * ---+6)) / +--(+-+-
7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0
 + ---(--+7 / (1 / +++6.37) + (1) /
482) / +++-+0)))) * -+5 + 7.513))))
- (+1 / ++((-84)))))))) * ++5 / +-(-
-2 - -++-9.0)))) / 5 * --++090 + * -
+5 + 7.513)))) - (+1 / ++((-84)))))))
)) * 8.2 - 27 - -9 / +((+9 * --2 + -
-+-+-((-1 * +(8 - 5 - 6)) * (-(a-+((
(+(4))))) - ++4) / +(-+---((5.6 - --
(3 * -1.8 * +(6 * +-(((-(-6) * ---+6
)) / +--(+-+-7 * (-0 * (+(((((2)) +
8 - 3 - ++9.0 + ---(--+7 / (1 / +++6
.37) + (1) / 482) / +++-+0)))) * -+5
 + 7.513)))) - (+1 / ++((-84)))))))))
 * ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090 ++5 / +-(--2 - -++-9.0)))) /
 5 * --++090
```

```
            stm.next()
            if expect_key:
                raise JSONError(E_DKEY, stm, stm.pos)
            if c == '}':
                return result
            expect_key = 1
            continue

        # parse out a key/value pair
        elif c == '"':
            key = _from_json_string(stm)
            stm.skipspaces()
            c = stm.next()
            if c != ':':
                raise JSONError(E_COLON, stm, stm.pos)
            stm.skipspaces()
            val = _from_json_raw(stm)
            result[key] = val
            expect_key = 0
            continue
       raise JSONError(E_MALF, stm, stm.pos)

def _from_json_raw(stm):
    while True:
        stm.skipspaces()
        c = stm.peek()
        if c == '"':
            return _from_json_string(stm)
        elif c == '{':
            return _from_json_dict(stm)
        elif c == '[':
            return _from_json_list(stm)
        elif c == 't':
            return _from_json_fixed(stm, 'true', True, E_BOOL)
        elif c == 'f':
            return _from_json_fixed(stm, 'false', False, E_BOOL)
        elif c == 'n':
            return _from_json_fixed(stm, 'null', None, E_NULL)
        elif c in NUMSTART:
            return _from_json_number(stm)
        raise JSONError(E_MALF, stm, stm.pos)

def from_json(data):
    stm = JSONStream(data)
    return _from_json_raw(stm)
```

microjson.py

```
<START>          ::= <json_raw>
<json_raw>       ::=  '"' <json_string'>
                    | '[' <json_list'>
                    | '{' <json_dict'>
                    | <json_number'>
                    | 'true'
                    | 'false'
                    | 'null'
<json_number'>   ::= <json_number>+
                    | <json_number>+ 'e' <json_number>+
<json_number>    ::=  '+' | '-' | '.' | [0-9] | 'E' | 'e'
<json_string'>   ::= <json_string>* '"'
<json_list'>     ::=  ']'
                    | <json_raw> (','<json_raw>)* ']'
                    | ( ',' <json_raw>)+ (',' <json_raw>)* ']'
<json_dict'>     ::=  '}'
                    | ( '"' <json_string'> ':' <json_raw> ',' )*
                      '"'<json_string'> ':' <json_raw> '}'
<json_string>    ::=  ' ' | '!' | '#' | '$' | '%' | '&' | '''
                    | '*' | '+' | '-' | ',' | '.' | '/' | ':' | ';'
                    | '<' | '=' | '>' | '?' | '@' | '[' | ']' | '^'
                    | '_', '`',|    '{' | '|' | '}' | '~'
                    | '[A-Za-z0-9]'
                    | '\' <decode_escape>
<decode_escape>  ::=  '"' | '/' | 'b' | 'f' | 'n' | 'r' | 't'
```

Recovered JSON grammar


nbviewer.org/github/vrthra/mimid/blob/master/src/PymimidBook.ipynb
jupyter
nbviewer
JUPYTER
FAQ
mimid / src
Mimid : Inferring Grammars
Code for subjects here
Evaluation starts here
The evaluation on specific subjects starts here
CGIDecode
Calculator
MathExpr
URLParse
Microjson
Lisp
Results are here
Recovering parse tree from a recognizer is here)
Recovering parse tree from parser combinators is here
Recovering parse tree from PEG parer is here
Please note that a complete run can take an hour and a half to complete.
We start with a few Jupyter magics that let us specify examples inline, that can be turned off if needed for faster execution. Switch TOP to False if you do not want examples to complete.
In [1]:
TOP = __name__ == '__main__'
The magics we use are %%var and %top . The %%var lets us specify large strings such as file contents directly without too many escapes. The %top helps with examples.
In [2]:
from IPython.core.magic import (Magics, magics_class, cell_magic, line_magic, line_cell_magic)
class B(dict):
def __getattr__(self, name):
return self.__getitem__(name)
@magics_class
class MyMagics(Magics):
def __init__(self, shell=None, **kwargs):
super().__init__(shell=shell, **kwargs)

We Found A Crash

We Found A Crash 

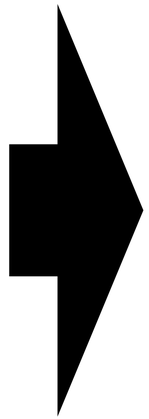
Why Did My Program Crash?

Why Did My Program Crash?

```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
 5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.
6 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +-
-(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) +
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.6
 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +--
(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(-
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) * -
+5 + 7.513)))) - (+1 / ++((-84))))))) * ++5 / +-
(--2 - -++-9.0)))) / 5 * --++090 + * -+5 + 7.513)
))) - (+1 / ++((-84))))))) * 8.2 - 27 - -9 / +((
+9 * --2 + --+-+-((-1 * +(8 - 5 - 6)) * (-(a-+
(((+(4))))) - ++4) / +(-+---((5.6 - --(3 * -1.8 *
 +(6 * +-(((-(-6) * ---+6)) / +--(+-+-7 * (-0 * (
+(((((2)) + 8 - 3 - ++9.0 + ---(--+7 / (1 / +++6.
37) + (1) / 482) / +++-+0)))) * -+5 + 7.513)))) -
(+1 / ++((-84))))))) * ++5 / +-(--2 - -++-9.0)))
) / 5 * --++090 ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090
```

Why Did My Program Crash?

```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
 5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.
6 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +-
-(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) +
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.6
 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +--
(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(-
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) * -
+5 + 7.513)))) - (+1 / ++((-84))))))) * ++5 / +-
(--2 - -++-9.0)))) / 5 * --++090 + * -+5 + 7.513)
))) - (+1 / ++((-84))))))) * 8.2 - 27 - -9 / +((
+9 * --2 + --+-+-((-1 * +(8 - 5 - 6)) * (-(a-+
(((+(4))))) - ++4) / +(-+---((5.6 - --(3 * -1.8 *
 +(6 * +-(((-(-6) * ---+6)) / +--(+-+-7 * (-0 * (
+(((((2)) + 8 - 3 - ++9.0 + ---(--+7 / (1 / +++6.
37) + (1) / 482) / +++-+0)))) * -+5 + 7.513)))) -
(+1 / ++((-84))))))) * ++5 / +-(--2 - -++-9.0)))
) / 5 * --++090 ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090
```

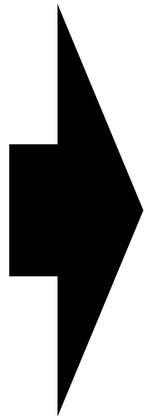


((4))

DD Minimized Input

Why Did My Program Crash?

```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
 5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.
6 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +-
-(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) +
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.6
 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +--
(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(-
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) * -
+5 + 7.513)))) - (+1 / ++((-84))))))) * ++5 / +-
(--2 - -++-9.0)))) / 5 * --++090 + * -+5 + 7.513)
))) - (+1 / ++((-84))))))) * 8.2 - 27 - -9 / +((
+9 * --2 + --+-+-((-1 * +(8 - 5 - 6)) * (-(a-+
(((+(4))))) - ++4) / +(-+---((5.6 - --(3 * -1.8 *
 +(6 * +-(((-(-6) * ---+6)) / +--(+-+-7 * (-0 * (
+(((((2)) + 8 - 3 - ++9.0 + ---(--+7 / (1 / +++6.
37) + (1) / 482) / +++-+0)))) * -+5 + 7.513)))) -
(+1 / ++((-84))))))) * ++5 / +-(--2 - -++-9.0)))
) / 5 * --++090 ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090
```



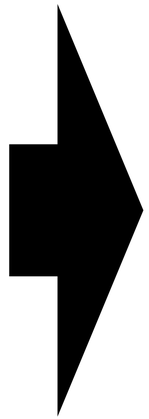
((4))

00000 ?

DD Minimized Input

Why Did My Program Crash?

```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
 5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.
6 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +-
-(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) +
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.6
 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +--
(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(-
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) * -
+5 + 7.513)))) - (+1 / ++((-84))))))) * ++5 / +-
(--2 - -++-9.0)))) / 5 * --++090 + * -+5 + 7.513)
))) - (+1 / ++((-84))))))) * 8.2 - 27 - -9 / +((
+9 * --2 + --+-+-((-1 * +(8 - 5 - 6)) * (-(a-+
(((+(4))))) - ++4) / +(-+---((5.6 - --(3 * -1.8 *
 +(6 * +-(((-(-6) * ---+6)) / +--(+-+-7 * (-0 * (
+(((((2)) + 8 - 3 - ++9.0 + ---(--+7 / (1 / +++6.
37) + (1) / 482) / +++-+0)))) * -+5 + 7.513)))) -
(+1 / ++((-84))))))) * ++5 / +-(--2 - -++-9.0)))
) / 5 * --++090 ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090
```



((4))

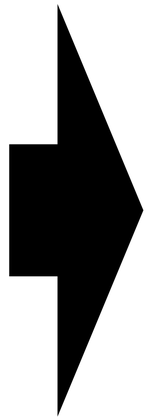
00000 ?

((5)) ?

DD Minimized Input

Why Did My Program Crash?

```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
 5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.
6 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +-
-(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) +
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 * +(8 -
5 - 6)) * (-(a-+(((+(4))))) - ++4) / +(-+---((5.6
 - --(3 * -1.8 * +(6 * +-(((-(-6) * ---+6)) / +--
(+-+-7 * (-0 * (+(((((2)) + 8 - 3 - ++9.0 + ---(-
--+7 / (1 / +++6.37) + (1) / 482) / +++-+0)))) * -
+5 + 7.513)))) - (+1 / ++((-84))))))) * ++5 / +-
(--2 - -++-9.0)))) / 5 * --++090 + * -+5 + 7.513)
))) - (+1 / ++((-84))))))) * 8.2 - 27 - -9 / +((
+9 * --2 + --+-+-((-1 * +(8 - 5 - 6)) * (-(a-+
(((+(4))))) - ++4) / +(-+---((5.6 - --(3 * -1.8 *
 +(6 * +-(((-(-6) * ---+6)) / +--(+-+-7 * (-0 * (
+(((((2)) + 8 - 3 - ++9.0 + ---(--+7 / (1 / +++6.
37) + (1) / 482) / +++-+0)))) * -+5 + 7.513)))) -
(+1 / ++((-84))))))) * ++5 / +-(--2 - -++-9.0)))
) / 5 * --++090 ++5 / +-(--2 - -++-9.0)))) / 5 *
--++090
```



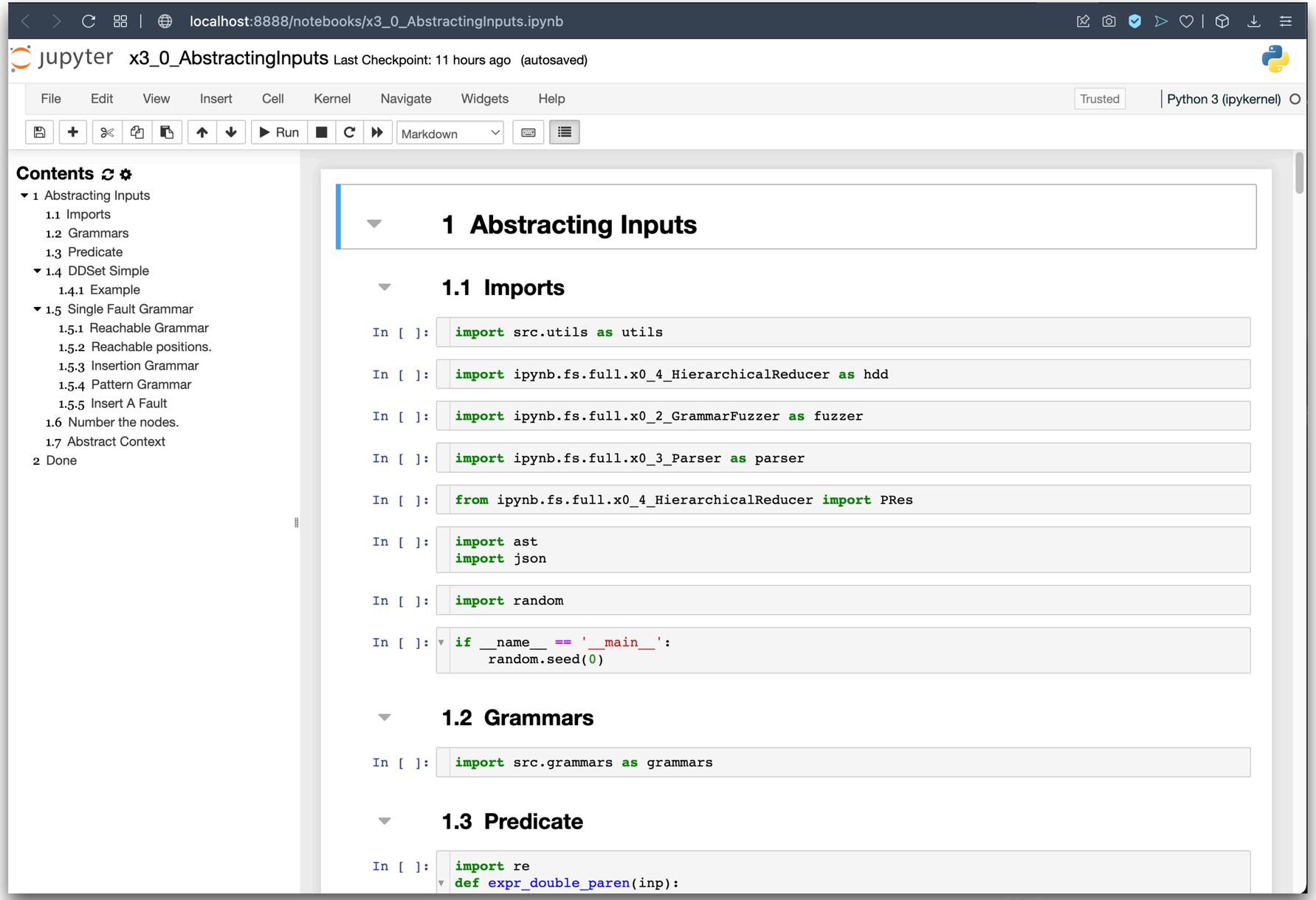
((4))

00000 ?

((5)) ?

(++5) ?

DD Minimized Input


localhost:8888/notebooks/x3_0_AbstractingInputs.ipynb
jupyter x3_0_AbstractingInputs Last Checkpoint: 11 hours ago (autosaved)
File Edit View Insert Cell Kernel Navigate Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
Contents
1 Abstracting Inputs
1.1 Imports
1.2 Grammars
1.3 Predicate
1.4 DDSet Simple
1.4.1 Example
1.5 Single Fault Grammar
1.5.1 Reachable Grammar
1.5.2 Reachable positions.
1.5.3 Insertion Grammar
1.5.4 Pattern Grammar
1.5.5 Insert A Fault
1.6 Number the nodes.
1.7 Abstract Context
2 Done
1 Abstracting Inputs
1.1 Imports
In [ ]: import src.utils as utils
In [ ]: import ipynb.fs.full.x0_4_HierarchicalReducer as hdd
In [ ]: import ipynb.fs.full.x0_2_GrammarFuzzer as fuzzer
In [ ]: import ipynb.fs.full.x0_3_Parser as parser
In [ ]: from ipynb.fs.full.x0_4_HierarchicalReducer import PRes
In [ ]: import ast
import json
In [ ]: import random
In [ ]: if __name__ == '__main__':
random.seed(0)
1.2 Grammars
In [ ]: import src.grammars as grammars
1.3 Predicate
In [ ]: import re
def expr_double_paren(inp):

```
var A = class  extends (class {}){};
```

Issue 2937 from Closure

```
const [y,y] = [];
```

Issue 386 from Rhino

```
var {baz:{} = baz => {}} = baz => {};
```

Issue 385 from Rhino

```
{while ((l_0)){ if ((l_0)) {break;;var l_0; continue }0}}
```

Issue 2842 from Closure

```
var A = class  extends (class {}){};
```

Issue 2937 from Closure

```
const [y,y] = [];
```

Issue 386 from Rhino

```
var {baz:{} = baz => {}} = baz => {};
```

Issue 385 from Rhino

```
{while ((l_0)){ if ((l_0)) {break;;var l_0; continue }0}}
```

Issue 2842 from Closure

```
var A = class  extends (class {}){};
```

Issue 2937 from Closure

```
const [y,y] = [];
```

Issue 386 from Rhino


```
var {baz:{} = baz => {}} = baz => {};
```

Issue 385 from Rhino

```
{while ((l_0)){ if ((l_0)) {break;;var l_0; continue }0}}
```

Issue 2842 from Closure

```
var A = class  extends (class {}){};
```

Issue 2937 from Closure

```
const [y,y] = [];
```

Issue 386 from Rhino

```
var {baz:{} = baz => {}} = baz => {};
```

Issue 385 from Rhino

```
{while ((l_0)){ if ((l_0)) {break;;var l_0; continue }0}}
```

Issue 2842 from Closure

```
var A = class  extends (class {}){};
```

Issue 2937 from Closure

```
const [y,y] = [];
```

Issue 386 from Rhino

```
var {baz:{} = baz => {}} = baz => {};
```

Issue 385 from Rhino

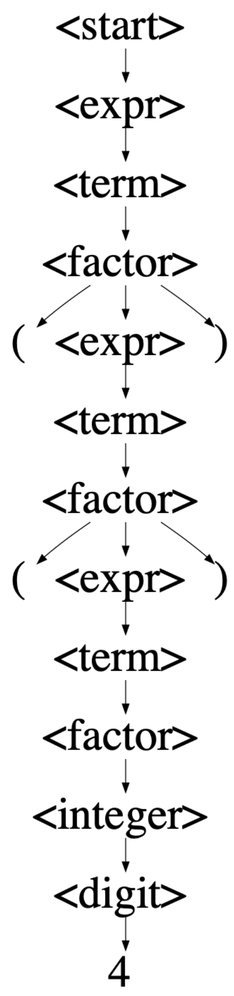
```
{while ((l_0)){ if ((l_0)) {break;;var l_0; continue }0}}
```

Issue 2842 from Closure

Delta Minimization is useful but not sufficient

((4))

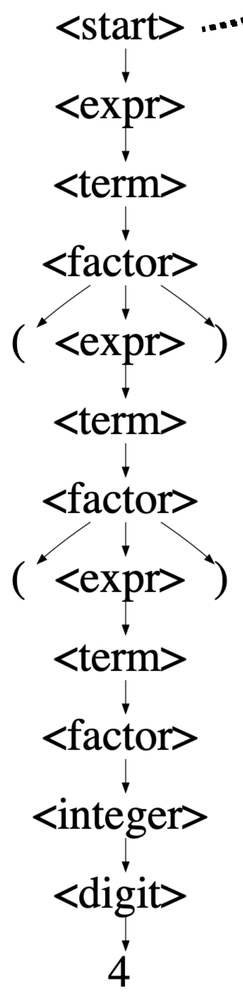
((4))



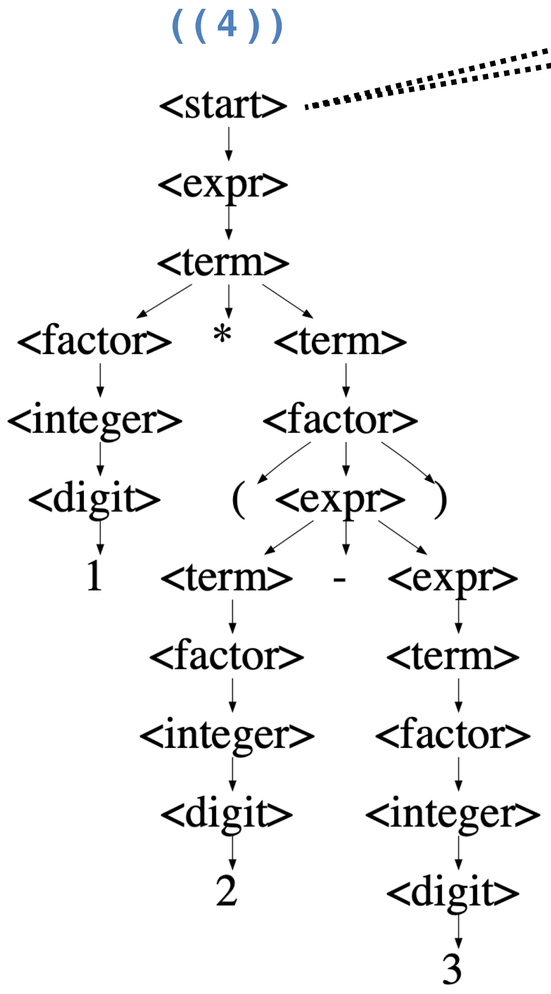

```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
 <term>   := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
 <factor> := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
 <integer>:= <digit> <integer>
           | <digit>
 <digit>  := [0-9]
```

((4))

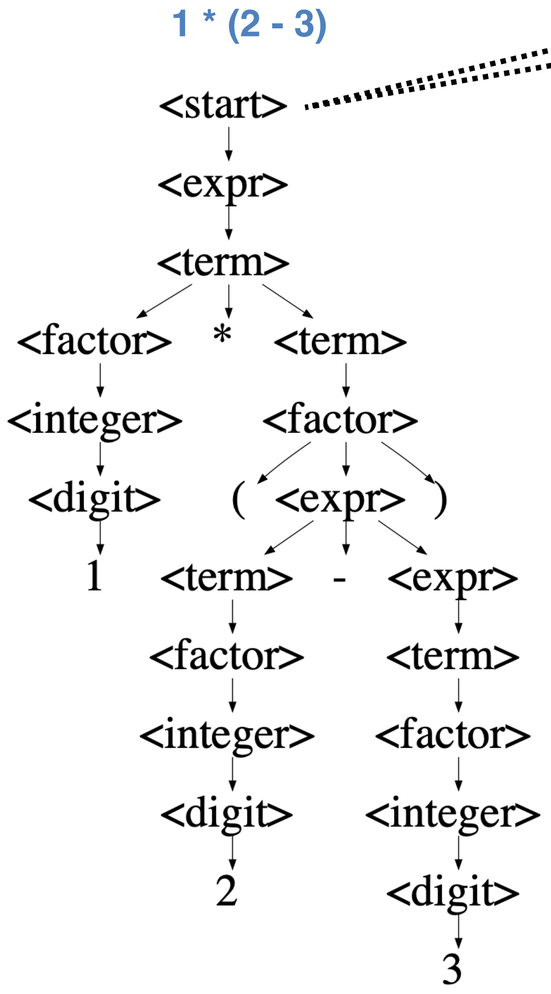
((4))




```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
 <term>   := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
 <factor> := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
 <integer>:= <digit> <integer>
           | <digit>
 <digit>  := [0-9]
```



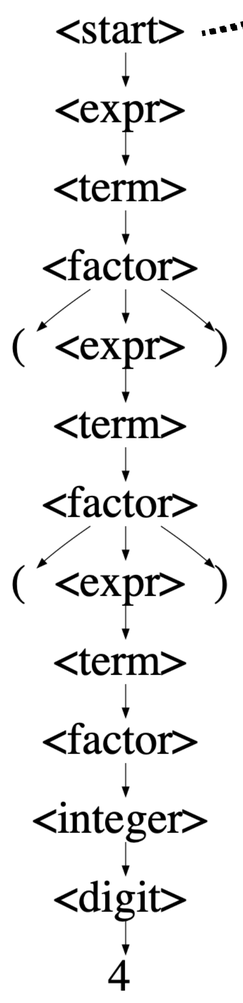

```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
 <term>   := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
 <factor> := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
 <integer>:= <digit> <integer>
           | <digit>
 <digit>  := [0-9]
```



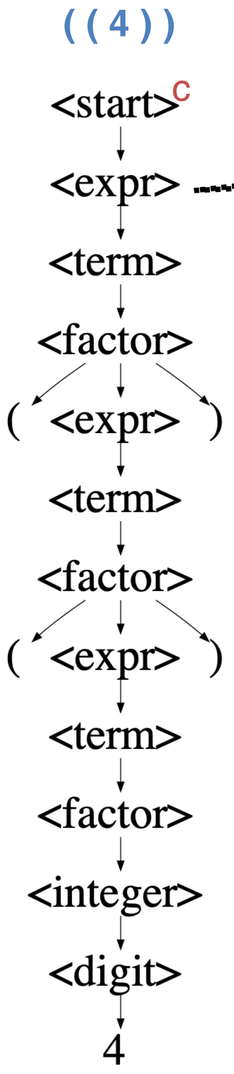

```
<start>  := <expr>

<expr>   := <term> ' + ' <expr>
          | <term> ' - ' <expr>
          | <term>
<term>   := <factor> ' * ' <term>
          | <factor> ' / ' <term>
          | <factor>
<factor> := '+' <factor>
          | '-' <factor>
          | '(' <expr> ')'
          | <integer> '.' <integer>
          | <integer>
<integer>:= <digit> <integer>
          | <digit>
<digit>  := [0-9]
```

✓ **Did not reproduce the failure**

((4))

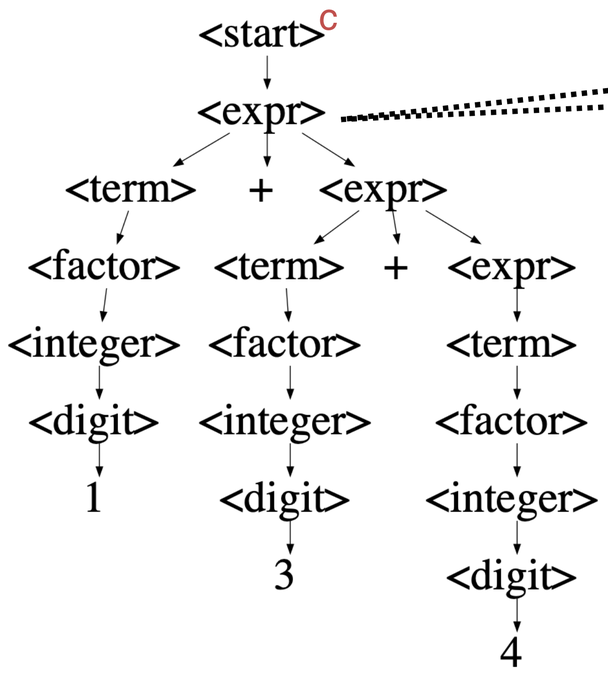



```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
 <term>   := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
 <factor> := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
 <integer>:= <digit> <integer>
           | <digit>
 <digit>  := [0-9]
```

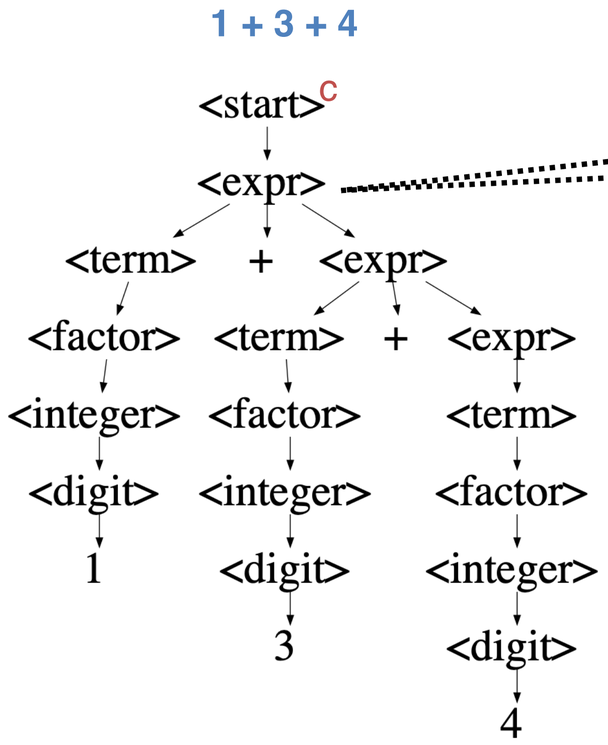



```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
<term>    := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
<integer> := <digit> <integer>
           | <digit>
<digit>   := [0-9]
```

((4))

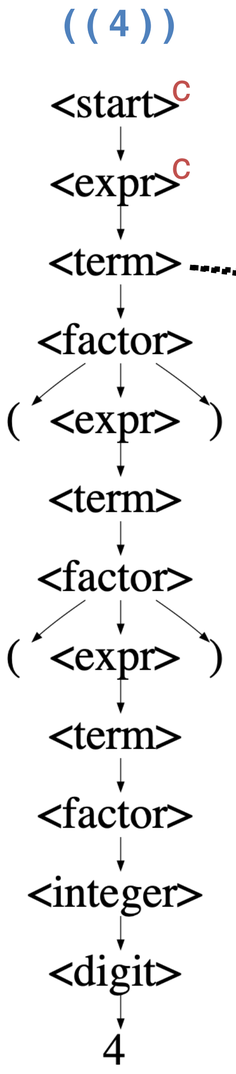



```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
<term>    := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
<integer> := <digit> <integer>
           | <digit>
<digit>   := [0-9]
```

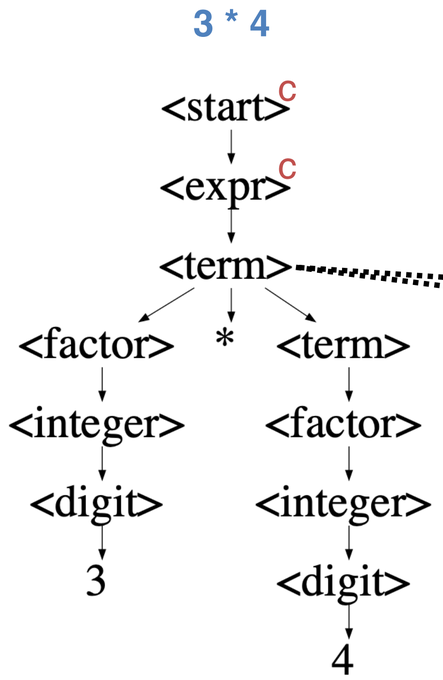



```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
<term>    := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
<integer> := <digit> <integer>
           | <digit>
<digit>   := [0-9]
```

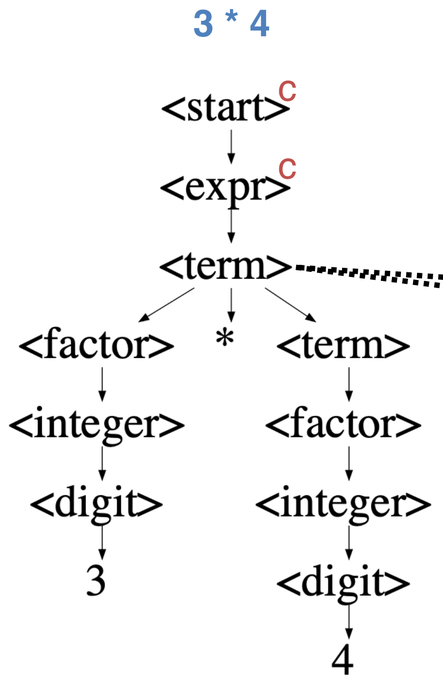
✓ **Did not reproduce the failure**




```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
 <term>   := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
<integer> := <digit> <integer>
           | <digit>
<digit>   := [0-9]
```

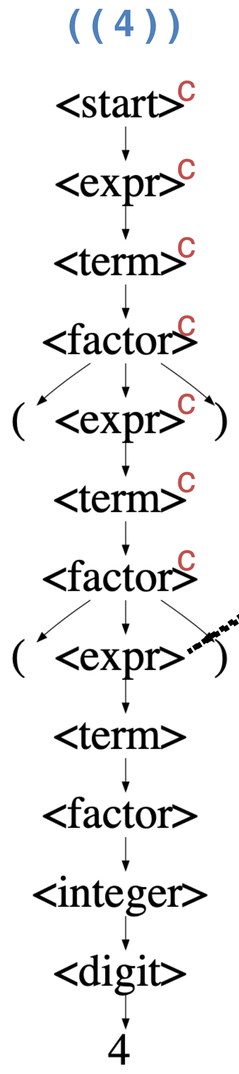



```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>

<term>    := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>

<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>

<integer> := <digit> <integer>
           | <digit>

<digit>   := [0-9]
```

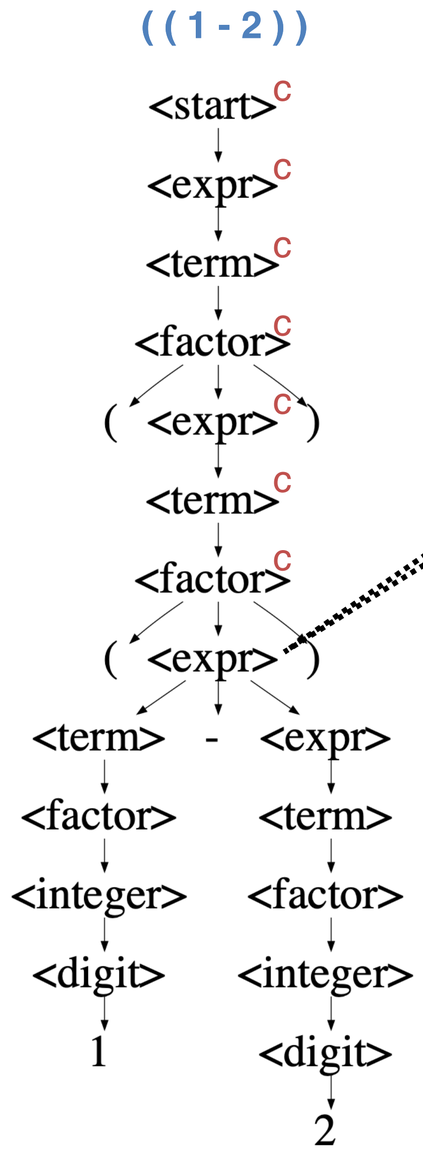



```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>

<term>    := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>

<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>

<integer> := <digit> <integer>
           | <digit>

<digit>   := [0-9]
```

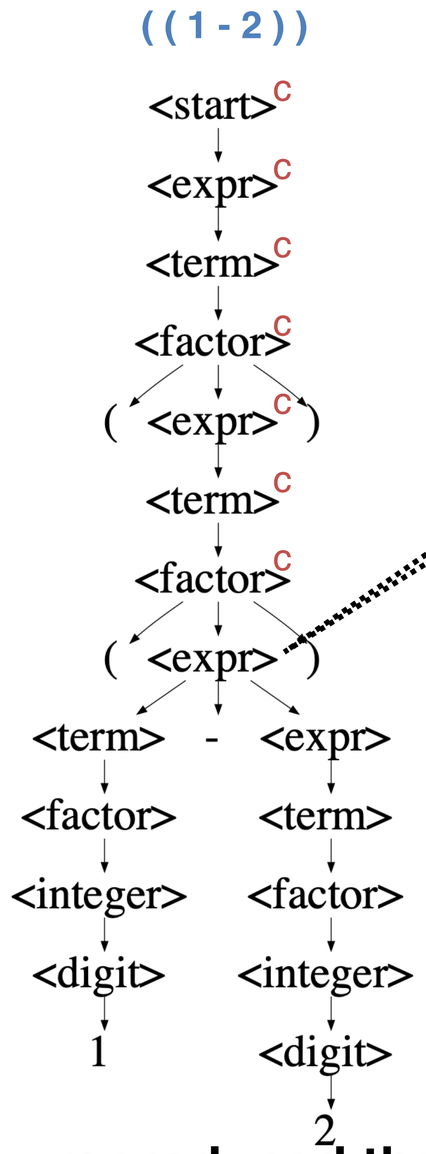
✓ **Did not reproduce the failure**




```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
<term>    := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
<integer> := <digit> <integer>
           | <digit>
<digit>   := [0-9]
```



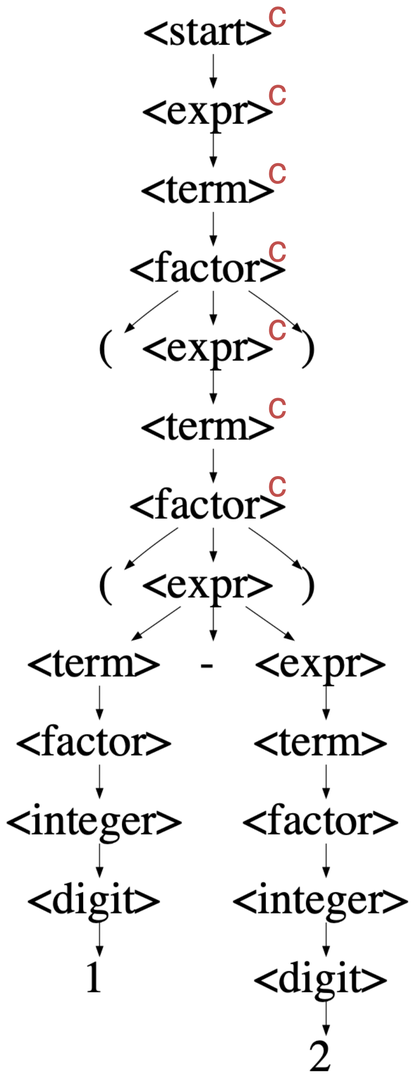

```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
<term>    := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
<integer> := <digit> <integer>
           | <digit>
<digit>   := [0-9]
```




```
<start>   := <expr>

<expr>    := <term> ' + ' <expr>
           | <term> ' - ' <expr>
           | <term>
<term>    := <factor> ' * ' <term>
           | <factor> ' / ' <term>
           | <factor>
<factor>  := '+' <factor>
           | '-' <factor>
           | '(' <expr> ')'
           | <integer> '.' <integer>
           | <integer>
<integer> := <digit> <integer>
           | <digit>
<digit>   := [0-9]
```

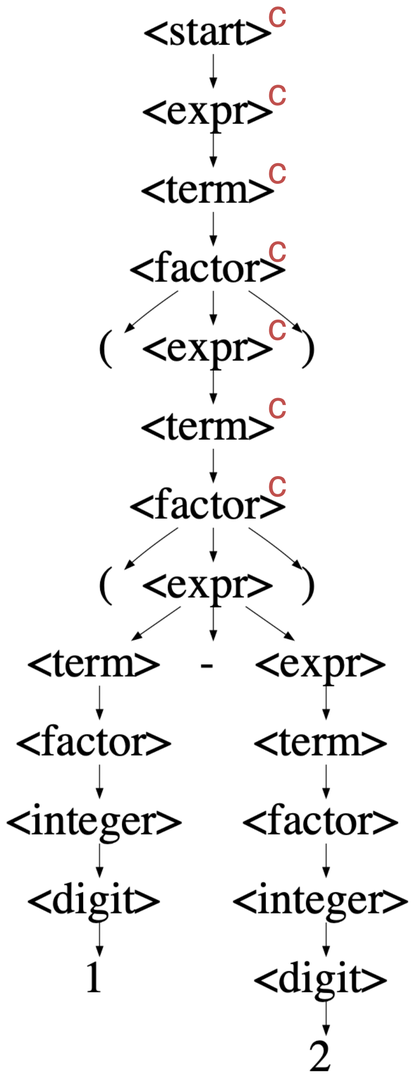
✘ **reproduced the failure**

((1-2))


<start>c
<expr>c
<term>c
<factor>c
(<expr>c)
<term>c
<factor>c
(<expr>)
<term> - <expr>
<factor> <term>
<integer> <factor>
<digit> <integer>
1 <digit>
2

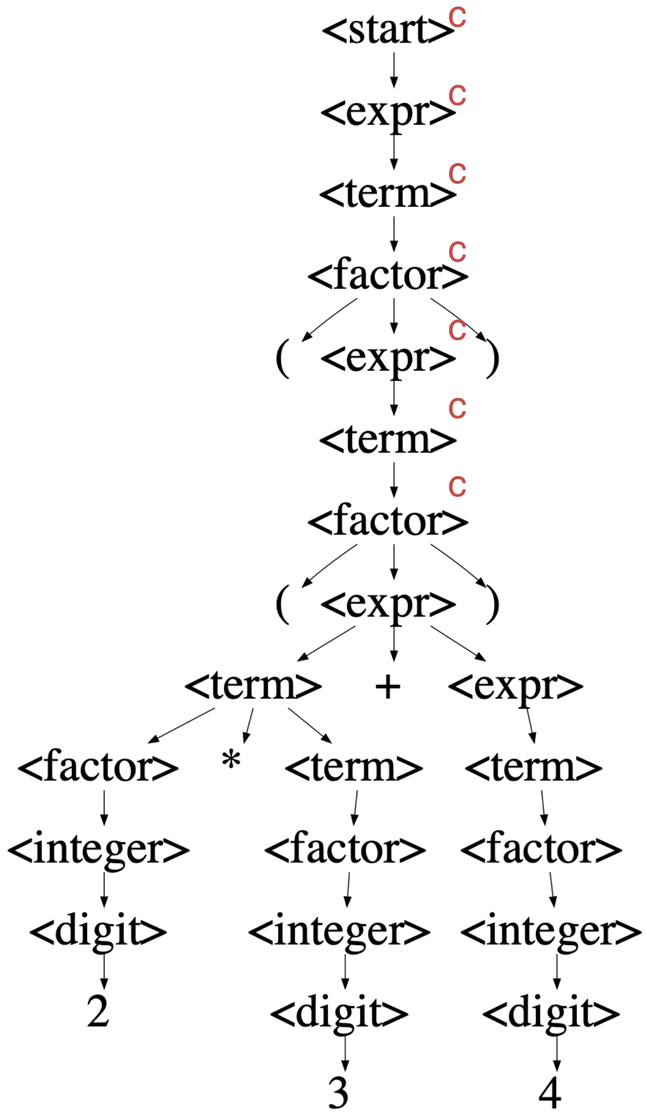
((1-2))

((1-2))


<start>C
<expr>C
<term>C
<factor>C
(<expr>C)
<term>C
<factor>C
(<expr>)
<term> - <expr>
<factor> <term>
<integer> <factor>
<digit> <integer>
1 <digit>
2

((1-2))

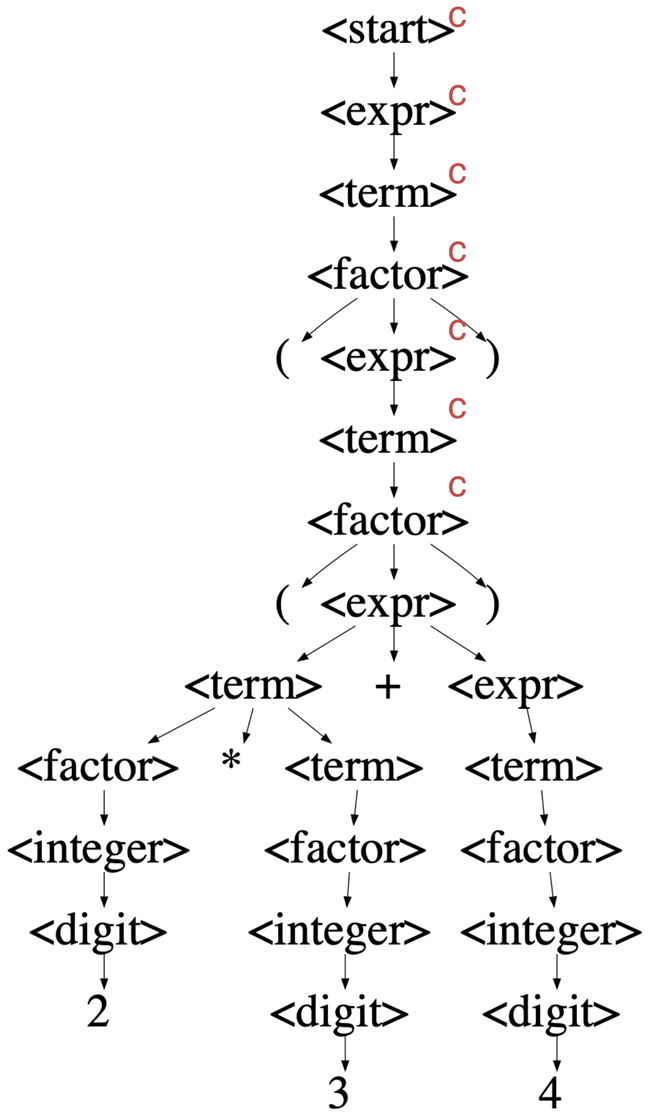
((1-2))


<start>
<expr>
<term>
<factor>
(<expr>)
<term>
<factor>
(<expr>)
<term> + <expr>
<factor> * <term> <term>
<integer> <factor> <factor>
<digit> <integer> <integer>
2 <digit> <digit>
3 4

((1-2))

((2*3+4))

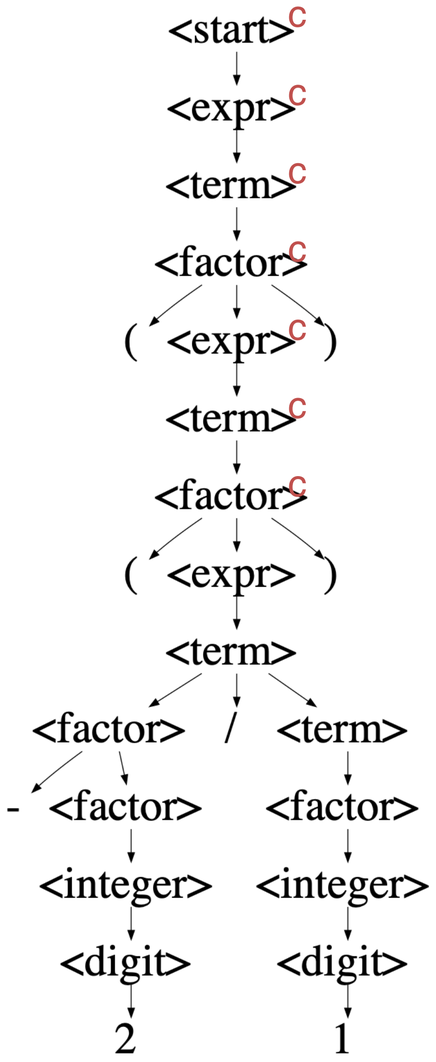
((1-2))


<start>
<expr>
<term>
<factor>
(<expr>)
<term>
<factor>
(<expr>)
<term> + <expr>
<factor> * <term> <term>
<integer> <factor> <factor>
<digit> <integer> <integer>
2 <digit> <digit>
3 4

((1-2)) ✗

((2*3+4)) ✗

((1-2))

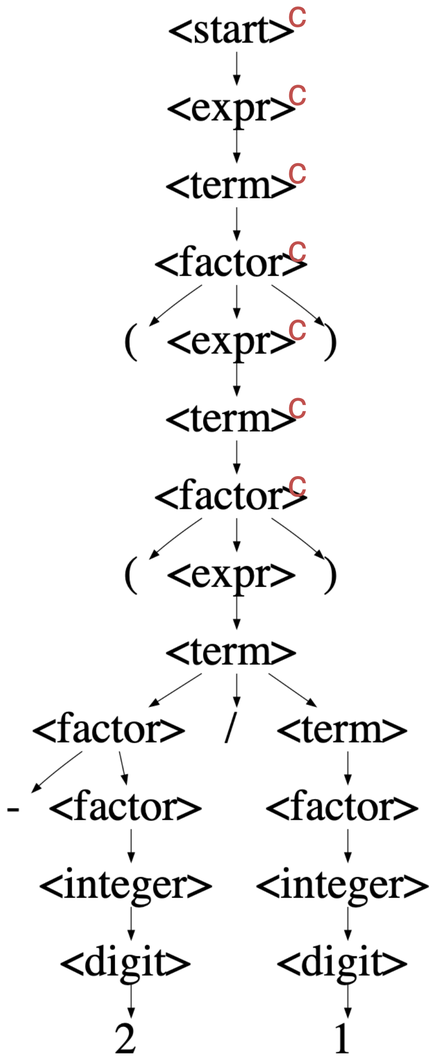

<start>C
<expr>C
<term>C
<factor>C
(<expr>C)
<term>C
<factor>C
(<expr>)
<term>
<factor> / <term>
- <factor> <factor>
<integer> <integer>
<digit> <digit>
2 1

((1-2)) ✗

((2*3+4)) ✗

((-2/1))

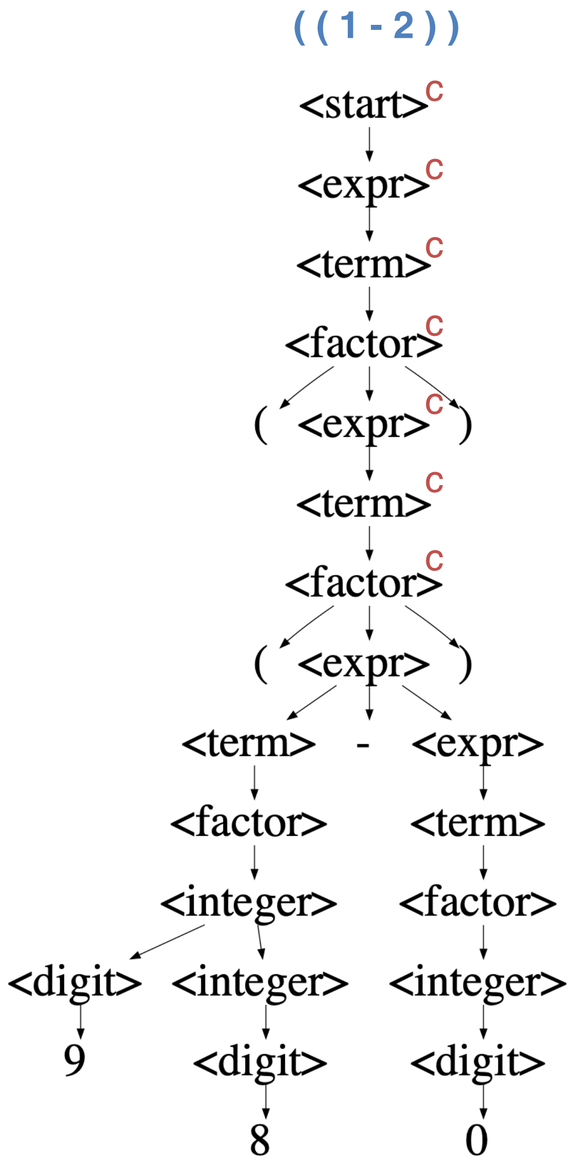
((1-2))


<start>C
<expr>C
<term>C
<factor>C
(<expr>C)
<term>C
<factor>C
(<expr>)
<term>
<factor> / <term>
- <factor> <factor>
<integer> <integer>
<digit> <digit>
2 1

((1-2)) ✗

((2*3+4)) ✗

((-2/1)) ✗

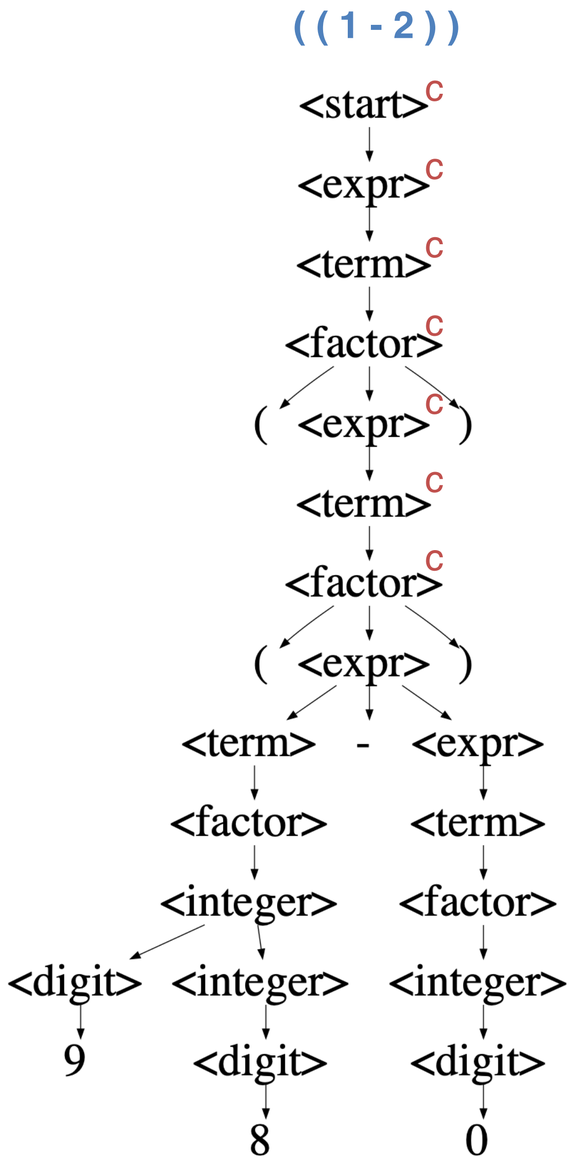

((1 - 2))
<start>C
<expr>C
<term>C
<factor>C
(<expr>C)
<term>C
<factor>C
(<expr>)
<term> - <expr>
<factor> <term>
<integer> <factor>
<digit> <integer> <integer>
9 <digit> <digit>
8 0

((1 - 2)) ✗

((2 * 3 + 4)) ✗

((- 2 / 1)) ✗

((98 - 0))


((1 - 2))
<start>C
<expr>C
<term>C
<factor>C
(<expr>C)
<term>C
<factor>C
(<expr>)
<term> - <expr>
<factor> <term>
<integer> <factor>
<digit> <integer> <integer>
9 <digit> <digit>
8 0

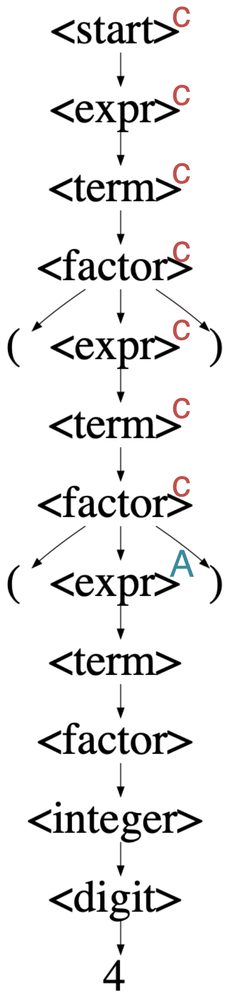
((1 - 2)) ✗

((2 * 3 + 4)) ✗

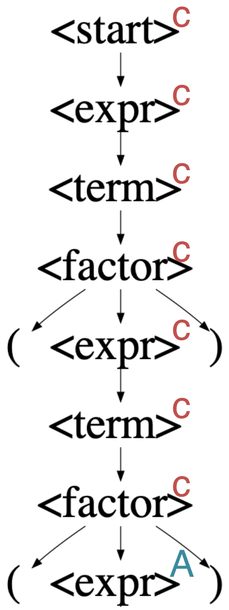
((- 2 / 1)) ✗

((98 - 0)) ✗

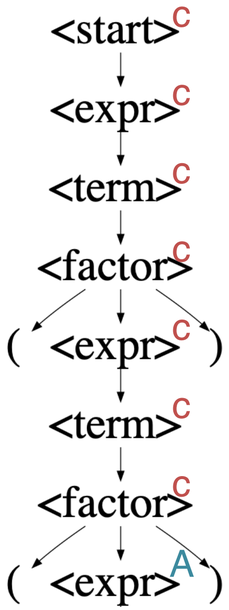
((4))


<start>C
<expr>C
<term>C
<factor>C
(<expr>C)
<term>C
<factor>C
(<expr>A)
<term>
<factor>
<integer>
<digit>
4

((4))



((4))




```
def check(parsed):
    if parsed.is_nested() and parsed.child.is_nested():
        raise Exception()
    return input
```

Minimized Input ((4))

Abstract Failure Inducing Input ((<expr>))

Specialized Grammar

```
<start F>   := <expr F>
<expr F>  := <term F> ' + ' <expr>
           | <term> ' + ' <expr F>
           | <term F> ' - ' <expr>
           | <term> ' - ' <expr F>
           | <term F>
<term F>  := <factor F> ' * ' <term>
           | <factor> ' * ' <term F>
           | <factor F> ' / ' <term>
           | <factor> ' / ' <term F>
           | <factor F>
<factor F>  := '+' <factor F>
           | '-' <factor F>
           | '(' <expr F> ')'
           | '(' <expr F1> ')'
<expr F1>  := <term F2>
<term F2>  := <factor F3>
<factor F3>:= '(' <expr> ')'
```

<factor F> **is** ((<expr>))

Specialized Grammar

```
<start F>   := <expr F>
<expr F>  := <term F> ' + ' <expr>
           | <term> ' + ' <expr F>
           | <term F> ' - ' <expr>
           | <term> ' - ' <expr F>
           | <term F>
<term F>  := <factor F> ' * ' <term>
           | <factor> ' * ' <term F>
           | <factor F> ' / ' <term>
           | <factor> ' / ' <term F>
           | <factor F>
<factor F>  := '+' <factor F>
           | '-' <factor F>
           | '(' <expr F> ')'
           | '(' <expr F1> ')'
<expr F1>  := <term F2>
<term F2>  := <factor F3>
<factor F3>:= '(' <expr> ')'
```

<factor F> **is** ((<expr>))

((1)) + 2

(23 * ((3)) - 34)

(344- 4 + ((223)))

(1) - 3 * 773 + (-22 + 1)

1798 - 889 / ((333-1)) * 2 / 3 + 1

34 + ((4)) - 2 (334 - (2 9 * 0 + 1

98435747+ 88)) + 7 / 4 * 889 - 2

8 + ((8)) + --1 + 112 + (1) - 456 + 134 / 45

437 + 8 - 1 * ((9 + ((1))) - 1 + 3 --1 + (112) - 2 + 445) + 0

74 + 334 + ((178 - 88 / (3393-1) + 1+ 3439)) * 223 - 1233 + 33467

2 * ((9)) - (1798 - 889 / (333-1) * 2 / 3 + 10 392 + 234 ----6 * 1798 - 889 / (33

778 - (((1) - 3 * 773 + (-22 + 1) * (4545) - 23 - ((2)) / 3434 + ---1 + 1 / 34343 + 112

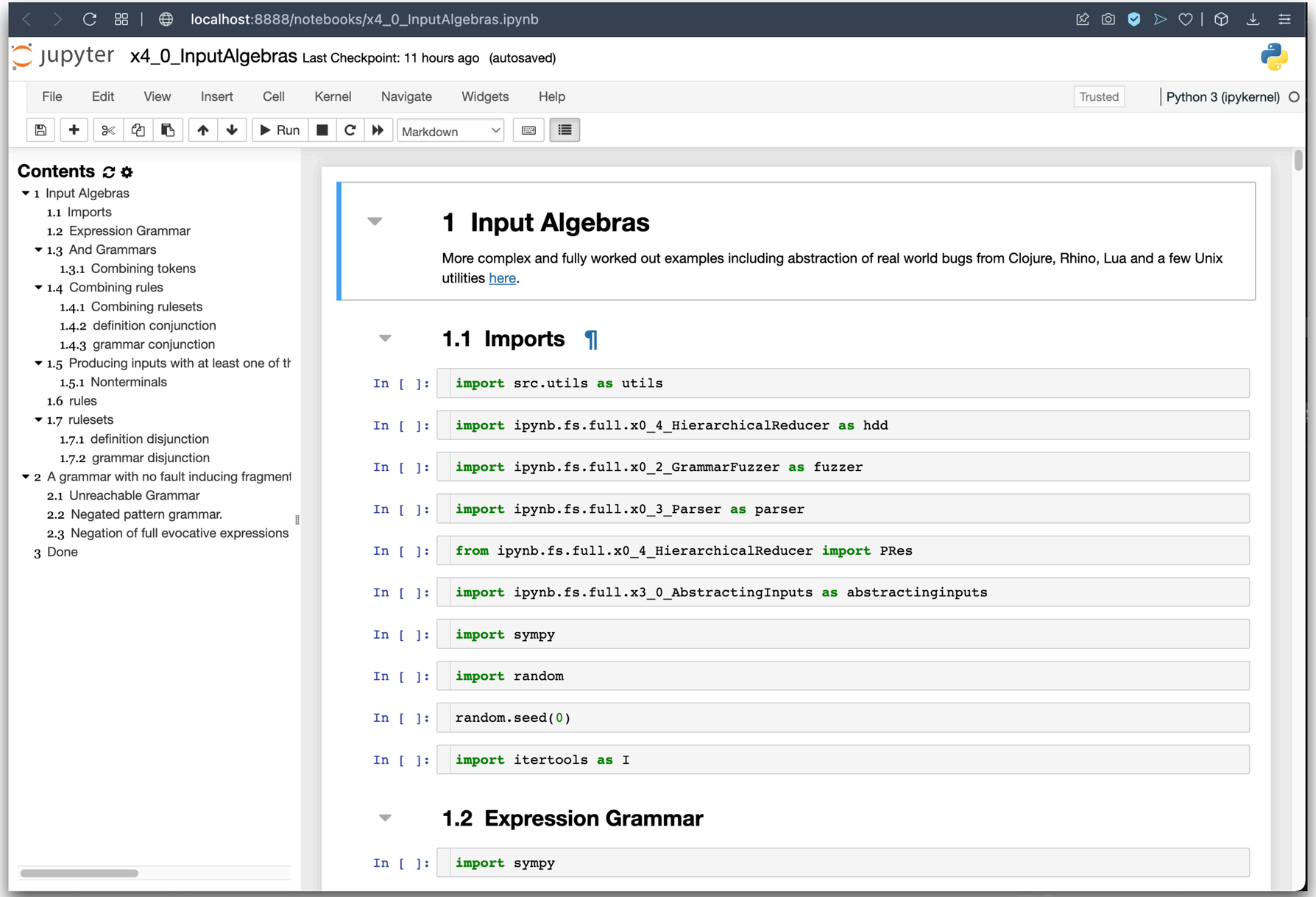
349 + (((1) - 3 * 3 + (-22 + 1) ((+ (-22 + 1) * (4545) - 23 - (2) ((-22 3434 + ---1 + 1 / 34343 + 1123

8 + ((8)) + --1 + / 1 - 39 + (1) - 456 + 134 / 45))(((1) - 2334 + ((((1)) - 3 + (-22 + 1) * (2) - 23 - (2) * 773 + (-22 + 1) / 3

74 + 3 + ((178 - 88 / (3393-1) * 1002 / 3 + 1+ 3439)) * - 1233 + 334672)) ((8 + ((8)) + --1 + / 344 - 39 + (1) - 456 + 134 / 45))(((1) - 3 * 77

1+ 33+ 24343433 +23343 - ((74 + 334 + ((178 - 88 / (3393-1) * 1002 / 3 + 1+ 3439)) * - 1233 + 334672)) ((8 + ((8)) + --1 + / 344 - 39 + (1) - 456 + 134 / 4

Input Algebras


localhost:8888/notebooks/x4_0_InputAlgebras.ipynb
jupyter x4_0_InputAlgebras Last Checkpoint: 11 hours ago (autosaved)
File Edit View Insert Cell Kernel Navigate Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
Contents
1 Input Algebras
1.1 Imports
1.2 Expression Grammar
1.3 And Grammars
1.3.1 Combining tokens
1.4 Combining rules
1.4.1 Combining rulesets
1.4.2 definition conjunction
1.4.3 grammar conjunction
1.5 Producing inputs with at least one of th
1.5.1 Nonterminals
1.6 rules
1.7 rulesets
1.7.1 definition disjunction
1.7.2 grammar disjunction
2 A grammar with no fault inducing fragment
2.1 Unreachable Grammar
2.2 Negated pattern grammar.
2.3 Negation of full evocative expressions
3 Done
1 Input Algebras
More complex and fully worked out examples including abstraction of real world bugs from Clojure, Rhino, Lua and a few Unix utilities here.
1.1 Imports ¶
In [ ]: import src.utils as utils
In [ ]: import ipynb.fs.full.x0_4_HierarchicalReducer as hdd
In [ ]: import ipynb.fs.full.x0_2_GrammarFuzzer as fuzzer
In [ ]: import ipynb.fs.full.x0_3_Parser as parser
In [ ]: from ipynb.fs.full.x0_4_HierarchicalReducer import PRes
In [ ]: import ipynb.fs.full.x3_0_AbstractingInputs as abstractinginputs
In [ ]: import sympy
In [ ]: import random
In [ ]: random.seed(0)
In [ ]: import itertools as I
1.2 Expression Grammar
In [ ]: import sympy

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| ‘true’ | ‘false’ | ‘null’
⟨*object*⟩ ::= ‘{’ ⟨*items*⟩ ‘}’ | ‘{}’
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ‘,’ ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ‘:’ ⟨*elt*⟩
⟨*array*⟩ ::= ‘[’ ⟨*elts*⟩ ‘]’ | ‘[]’
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ‘,’ ⟨*elts*⟩
⟨*string*⟩ ::= ‘"’ ⟨*chars*⟩ ‘"’
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= ‘[A-Za-z0-9]’
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= ‘[0-9]’

```
def jsoncheck(json):
  ...
```

{...}

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| ‘true’ | ‘false’ | ‘null’
⟨*object*⟩ ::= ‘{’ ⟨*items*⟩ ‘}’ | ‘{}’
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ‘,’ ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ‘:’ ⟨*elt*⟩
⟨*array*⟩ ::= ‘[’ ⟨*elts*⟩ ‘]’ | ‘[]’
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ‘,’ ⟨*elts*⟩
⟨*string*⟩ ::= ‘"’ ⟨*chars*⟩ ‘"’
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= ‘[A-Za-z0-9]’
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= ‘[0-9]’

```
def jsoncheck(json):
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

{"abc": null} ✗

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| 'true' | 'false' | 'null'
⟨*object*⟩ ::= '{' ⟨*items*⟩ '}' | '{}'
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ',' ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ':' ⟨*elt*⟩
⟨*array*⟩ ::= '[' ⟨*elts*⟩ ']' | '[]'
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ',' ⟨*elts*⟩
⟨*string*⟩ ::= '"' ⟨*chars*⟩ '"'
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= '[A-Za-z0-9]'
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= '[0-9]'

```
def jsoncheck(json):
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

<item N> **is** <string> : null

{"abc": null} ✗

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| ‘true’ | ‘false’ | ‘null’
⟨*object*⟩ ::= ‘{’ ⟨*items*⟩ ‘}’ | ‘{}’
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ‘,’ ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ‘:’ ⟨*elt*⟩
⟨*array*⟩ ::= ‘[’ ⟨*elts*⟩ ‘]’ | ‘[]’
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ‘,’ ⟨*elts*⟩
⟨*string*⟩ ::= ‘"’ ⟨*chars*⟩ ‘"’
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= ‘[A-Za-z0-9]’
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= ‘[0-9]’

```
def jsoncheck(json):
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

{"abc": []} ✔

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| ‘true’ | ‘false’ | ‘null’
⟨*object*⟩ ::= ‘{’ ⟨*items*⟩ ‘}’ | ‘{}’
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ‘,’ ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ‘:’ ⟨*elt*⟩
⟨*array*⟩ ::= ‘[’ ⟨*elts*⟩ ‘]’ | ‘[]’
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ‘,’ ⟨*elts*⟩
⟨*string*⟩ ::= ‘"’ ⟨*chars*⟩ ‘"’
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= ‘[A-Za-z0-9]’
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= ‘[0-9]’

```
def jsoncheck(json):
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

no *<item N>* **is** <string> : null

{"abc": []} ✔

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| ‘true’ | ‘false’ | ‘null’
⟨*object*⟩ ::= ‘{’ ⟨*items*⟩ ‘}’ | ‘{}’
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ‘,’ ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ‘:’ ⟨*elt*⟩
⟨*array*⟩ ::= ‘[’ ⟨*elts*⟩ ‘]’ | ‘[]’
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ‘,’ ⟨*elts*⟩
⟨*string*⟩ ::= ‘"’ ⟨*chars*⟩ ‘"’
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= ‘[A-Za-z0-9]’
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= ‘[0-9]’

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  process(json)
```

no *<item E>* **is** "" : *<elt>*

{"abc": 124} ✗

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| ‘true’ | ‘false’ | ‘null’
⟨*object*⟩ ::= ‘{’ ⟨*items*⟩ ‘}’ | ‘{}’
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ‘,’ ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ‘:’ ⟨*elt*⟩
⟨*array*⟩ ::= ‘[’ ⟨*elts*⟩ ‘]’ | ‘[]’
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ‘,’ ⟨*elts*⟩
⟨*string*⟩ ::= ‘"’ ⟨*chars*⟩ ‘"’
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= ‘[A-Za-z0-9]’
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= ‘[0-9]’

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  process(json)
```

<item E> **is** "" : *<elt>*

{"": 124}

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| ‘true’ | ‘false’ | ‘null’
⟨*object*⟩ ::= ‘{’ ⟨*items*⟩ ‘}’ | ‘{}’
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ‘,’ ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ‘:’ ⟨*elt*⟩
⟨*array*⟩ ::= ‘[’ ⟨*elts*⟩ ‘]’ | ‘[]’
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ‘,’ ⟨*elts*⟩
⟨*string*⟩ ::= ‘"’ ⟨*chars*⟩ ‘"’
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= ‘[A-Za-z0-9]’
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= ‘[0-9]’

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

⟨*json*⟩ ::= ⟨*elt*⟩
⟨*elt*⟩ ::= ⟨*object*⟩ | ⟨*array*⟩ | ⟨*string*⟩ | ⟨*number*⟩
| ‘true’ | ‘false’ | ‘null’
⟨*object*⟩ ::= ‘{’ ⟨*items*⟩ ‘}’ | ‘{}’
⟨*items*⟩ ::= ⟨*item*⟩ | ⟨*item*⟩ ‘,’ ⟨*items*⟩
⟨*item*⟩ ::= ⟨*string*⟩ ‘:’ ⟨*elt*⟩
⟨*array*⟩ ::= ‘[’ ⟨*elts*⟩ ‘]’ | ‘[]’
⟨*elts*⟩ ::= ⟨*elt*⟩ | ⟨*elt*⟩ ‘,’ ⟨*elts*⟩
⟨*string*⟩ ::= ‘"’ ⟨*chars*⟩ ‘"’
⟨*chars*⟩ ::= ⟨*char*⟩ ⟨*chars*⟩ | ϵ
⟨*char*⟩ ::= ‘[A-Za-z0-9]’
⟨*number*⟩ ::= ⟨*digits*⟩
⟨*digits*⟩ ::= ⟨*digit*⟩ ⟨*digits*⟩ | ⟨*digit*⟩
⟨*digit*⟩ ::= ‘[0-9]’

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

<item E> **is** "" : *<elt>*

&

no *<item N>* **is** <string> : null **{"": 124}**

Start Symbol

$\langle json_N \rangle$ ::= $\langle elt_N \rangle$
$\langle elt_N \rangle$ ::= $\langle object_N \rangle$ | $\langle array_N \rangle$
$\langle array_N \rangle$ ::= '[' $\langle elts_N \rangle$ ']'
$\langle object_N \rangle$ ::= '{' $\langle items_N \rangle$ '}'
$\langle elts_N \rangle$ ::= $\langle elt_N \rangle$ ',' $\langle elts \rangle$ | $\langle elt \rangle$ ',' $\langle elts_N \rangle$ | $\langle elt_N \rangle$
$\langle items_N \rangle$ ::= $\langle item_N \rangle$ ',' $\langle items \rangle$ | $\langle item \rangle$ ',' $\langle items_N \rangle$
| $\langle item\ N \rangle$
$\langle item_N \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_N \rangle$ | $\langle string \rangle$ ':' $\langle elt_{N_1} \rangle$
$\langle elt_{N_1} \rangle$ ::= 'null'

```
def jsoncheck(json):
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

<item N> **is** <string> : null

{"abc": null} ✗

$\langle json_{\overline{N}} \rangle$ ::= $\langle elt_{\overline{N}} \rangle$

$\langle elt_{\overline{N}} \rangle$ ::= 'false' | 'null' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle array_{\overline{N}} \rangle$ | $\langle object_{\overline{N}} \rangle$

$\langle array_{\overline{N}} \rangle$ ::= '[]' | '[' $\langle elts_{\overline{N}} \rangle$ ']'

$\langle object_{\overline{N}} \rangle$ ::= '{}' | '{' $\langle items_{\overline{N}} \rangle$ '}'

$\langle elts_{\overline{N}} \rangle$ ::= $\langle elt_{\overline{N}} \rangle$ | $\langle elt_{\overline{N}} \rangle$ ',' $\langle elts_{\overline{N}} \rangle$

$\langle items_{\overline{N}} \rangle$ ::= $\langle item_{\overline{N}} \rangle$ | $\langle item_{\overline{N}} \rangle$ ',' $\langle items_{\overline{N}} \rangle$

$\langle item_{\overline{N}} \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$

$\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$ ::= 'false' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle array_{\overline{N}} \rangle$ | $\langle object_{\overline{N}} \rangle$

Start Symbol

$\langle json_N \rangle$ ::= $\langle elt_N \rangle$

$\langle elt_N \rangle$ ::= $\langle object_N \rangle$ | $\langle array_N \rangle$

$\langle array_N \rangle$ ::= '[' $\langle elts_N \rangle$ ']'

$\langle object_N \rangle$ ::= '{' $\langle items_N \rangle$ '}'

$\langle elts_N \rangle$ ::= $\langle elt_N \rangle$ ',' $\langle elts \rangle$ | $\langle elt \rangle$ ',' $\langle elts_N \rangle$ | $\langle elt_N \rangle$

$\langle items_N \rangle$ ::= $\langle item_N \rangle$ ',' $\langle items \rangle$ | $\langle item \rangle$ ',' $\langle items_N \rangle$
| $\langle item\ N \rangle$

$\langle item_N \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_N \rangle$ | $\langle string \rangle$ ':' $\langle elt_{N_1} \rangle$

$\langle elt_{N_1} \rangle$ ::= 'null'

```
def jsoncheck(json):
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

no *<item N>* **is** <string> : null

{"abc": []} ✔

Start Symbol

$\langle json_E \rangle$ ::= $\langle elt_E \rangle$
$\langle elt_E \rangle$ ::= $\langle object_E \rangle$ | $\langle array_E \rangle$
$\langle object_E \rangle$ ::= '{' $\langle items_E \rangle$ '}'
$\langle items_E \rangle$ ::= $\langle item_E \rangle$ ',' $\langle items \rangle$ | $\langle item \rangle$ ',' $\langle items_E \rangle$
| $\langle item_E \rangle$
$\langle item_E \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_E \rangle$ | $\langle string_{E_1} \rangle$ ':' $\langle elt \rangle$
$\langle array_E \rangle$ ::= '[' $\langle elts_E \rangle$ ']'
$\langle elts_E \rangle$ ::= $\langle elt_E \rangle$ ',' $\langle elts \rangle$ | $\langle elt \rangle$ ',' $\langle elts_E \rangle$ | $\langle elt_E \rangle$
$\langle string_{E_1} \rangle$ ::= '"' $\langle chars_{E_2} \rangle$ '"'
$\langle chars_{E_2} \rangle$ ::= ϵ

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  process(json)
```

<item E> **is** "" : *<elt>*

{"abc": 124} ✗

Start Symbol

$\langle json_{E\wedge\overline{N}} \rangle$ ::= $\langle elt_{E\wedge\overline{N}} \rangle$

$\langle elt_{E\wedge\overline{N}} \rangle$ ::= $\langle array_{E\wedge\overline{N}} \rangle$ | $\langle object_{E\wedge\overline{N}} \rangle$

$\langle array_{E\wedge\overline{N}} \rangle$ ::= '[' $\langle elts_{E\wedge\overline{N}} \rangle$ ']'

$\langle object_{E\wedge\overline{N}} \rangle$ ::= '{' $\langle items_{E\wedge\overline{N}} \rangle$ '}'

$\langle elts_{E\wedge\overline{N}} \rangle$ ::= $\langle elt_{E\wedge\overline{N}} \rangle$ | $\langle elt_{E\wedge\overline{N}} \rangle$ ',' $\langle elts_{\overline{N}} \rangle$
| $\langle elt_{\overline{N}} \rangle$ ',' $\langle elts_{E\wedge\overline{N}} \rangle$

$\langle elt_{\overline{N}\wedge\overline{N_1}} \rangle$ ::= 'false' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle object_{\overline{N}} \rangle$ | $\langle array_{\overline{N}} \rangle$

$\langle items_{E\wedge\overline{N}} \rangle$ ::= $\langle item_{E\wedge\overline{N}} \rangle$ | $\langle item_{E\wedge\overline{N}} \rangle$ ',' $\langle items_{\overline{N}} \rangle$
| $\langle item_{\overline{N}} \rangle$ ',' $\langle items_{E\wedge\overline{N}} \rangle$

$\langle item_{E\wedge\overline{N}} \rangle$ ::= $\langle string_{E_1} \rangle$ ':' $\langle elt_{\overline{N}\wedge\overline{N_1}} \rangle$
| $\langle string \rangle$ ':' $\langle elt_{E\wedge\overline{N}\wedge\overline{N_1}} \rangle$

$\langle elt_{E\wedge\overline{N}\wedge\overline{N_1}} \rangle$ ::= $\langle array_{E\wedge\overline{N}} \rangle$ | $\langle object_{E\wedge\overline{N}} \rangle$

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

<item E> **is** "" : *<elt>*

&

no *<item N>* **is** <string> : null **{"": 124}** ✔

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

Evogram

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

Evogram

$\langle json_{E \wedge \overline{N}} \rangle$

where $\langle item_E \rangle$ **is** `" "` : $\langle elt \rangle$

$\langle item_N \rangle$ **is** $\langle string \rangle$: `null`

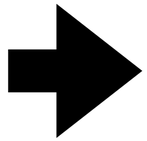
```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

Evogram

$$\langle json_{E \wedge \overline{N}} \rangle$$

where $\langle item_{E} \rangle$ **is** `" "` `:` $\langle elt \rangle$

$\langle item_{N} \rangle$ **is** $\langle string \rangle$ `:null`



Automatically Derived

$\langle json_{E \wedge \overline{N}} \rangle$ ::= $\langle elt_{E \wedge \overline{N}} \rangle$

$\langle elt_{E \wedge \overline{N}} \rangle$ ::= $\langle array_{E \wedge \overline{N}} \rangle$ | $\langle object_{E \wedge \overline{N}} \rangle$

$\langle array_{E \wedge \overline{N}} \rangle$ ::= '[' $\langle elts_{E \wedge \overline{N}} \rangle$ ']'

$\langle object_{E \wedge \overline{N}} \rangle$ ::= '{' $\langle items_{E \wedge \overline{N}} \rangle$ '}'

$\langle elts_{E \wedge \overline{N}} \rangle$ ::= $\langle elt_{E \wedge \overline{N}} \rangle$ | $\langle elt_{E \wedge \overline{N}} \rangle$ ',' $\langle elts_{\overline{N}} \rangle$
| $\langle elt_{\overline{N}} \rangle$ ',' $\langle elts_{E \wedge \overline{N}} \rangle$

$\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$ ::= 'false' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle object_{\overline{N}} \rangle$ | $\langle array_{\overline{N}} \rangle$

$\langle items_{E \wedge \overline{N}} \rangle$ ::= $\langle item_{E \wedge \overline{N}} \rangle$ | $\langle item_{E \wedge \overline{N}} \rangle$ ',' $\langle items_{\overline{N}} \rangle$
| $\langle item_{\overline{N}} \rangle$ ',' $\langle items_{E \wedge \overline{N}} \rangle$

$\langle item_{E \wedge \overline{N}} \rangle$ ::= $\langle string_{E_1} \rangle$ ':' $\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$
| $\langle string \rangle$ ':' $\langle elt_{E \wedge \overline{N} \wedge \overline{N_1}} \rangle$

$\langle elt_{E \wedge \overline{N} \wedge \overline{N_1}} \rangle$ ::= $\langle array_{E \wedge \overline{N}} \rangle$ | $\langle object_{E \wedge \overline{N}} \rangle$

$\langle json_{\overline{N}} \rangle$ ::= $\langle elt_{\overline{N}} \rangle$

$\langle elt_{\overline{N}} \rangle$ ::= 'false' | 'null' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle array_{\overline{N}} \rangle$ | $\langle object_{\overline{N}} \rangle$

$\langle array_{\overline{N}} \rangle$ ::= '[]' | '[' $\langle elts_{\overline{N}} \rangle$ ']'

$\langle object_{\overline{N}} \rangle$ ::= '{}' | '{' $\langle items_{\overline{N}} \rangle$ '}'

$\langle elts_{\overline{N}} \rangle$ ::= $\langle elt_{\overline{N}} \rangle$ | $\langle elt_{\overline{N}} \rangle$ ',' $\langle elts_{\overline{N}} \rangle$

$\langle items_{\overline{N}} \rangle$ ::= $\langle item_{\overline{N}} \rangle$ | $\langle item_{\overline{N}} \rangle$ ',' $\langle items_{\overline{N}} \rangle$

$\langle item_{\overline{N}} \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$

$\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$ ::= 'false' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle array_{\overline{N}} \rangle$ | $\langle object_{\overline{N}} \rangle$

$\langle json_{N} \rangle$ ::= $\langle elt_{N} \rangle$

$\langle elt_{N} \rangle$ ::= $\langle object_{N} \rangle$ | $\langle array_{N} \rangle$

$\langle array_{N} \rangle$ ::= '[' $\langle elts_{N} \rangle$ ']'

$\langle object_{N} \rangle$ ::= '{' $\langle items_{N} \rangle$ '}'

$\langle elts_{N} \rangle$ ::= $\langle elt_{N} \rangle$ ',' $\langle elts \rangle$ | $\langle elt \rangle$ ',' $\langle elts_{N} \rangle$ | $\langle elt_{N} \rangle$

$\langle items_{N} \rangle$ ::= $\langle item_{N} \rangle$ ',' $\langle items \rangle$ | $\langle item \rangle$ ',' $\langle items_{N} \rangle$
| $\langle item\ N \rangle$

$\langle item_{N} \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_{N} \rangle$ | $\langle string \rangle$ ':' $\langle elt_{N_1} \rangle$

$\langle elt_{N_1} \rangle$ ::= 'null'

$\langle json_{E} \rangle$ ::= $\langle elt_{E} \rangle$

$\langle elt_{E} \rangle$ ::= $\langle object_{E} \rangle$ | $\langle array_{E} \rangle$

$\langle object_{E} \rangle$ ::= '{' $\langle items_{E} \rangle$ '}'

$\langle items_{E} \rangle$ ::= $\langle item_{E} \rangle$ ',' $\langle items \rangle$ | $\langle item \rangle$ ',' $\langle items_{E} \rangle$
| $\langle item_{E} \rangle$

$\langle item_{E} \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_{E} \rangle$

$\langle array_{E} \rangle$ ::= '[' $\langle elts_{E} \rangle$ ']'

$\langle elts_{E} \rangle$ ::= $\langle elt_{E} \rangle$ ',' $\langle elts \rangle$ | $\langle elt \rangle$ ',' $\langle elts_{E} \rangle$ | $\langle elt_{E} \rangle$

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

$$\langle json_{E \wedge \overline{N}} \rangle$$

where $\langle item_E \rangle$ **is** `" "` : $\langle elt \rangle$

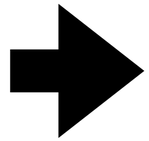
$\langle item_N \rangle$ **is** $\langle string \rangle$: `null`

```
def jsoncheck(json):
  if no_key_is_empty_string(json):
    fail('one key must be empty')
  if any_key_has_null_value(json):
    fail('key value must not be null')
  process(json)
```

$\langle json_{E \wedge \overline{N}} \rangle$

where $\langle item_E \rangle$ **is** `" "` : $\langle elt \rangle$

$\langle item_N \rangle$ **is** $\langle string \rangle$: `null`



Automatically Derived

$\langle json_{E \wedge \overline{N}} \rangle$ ::= $\langle elt_{E \wedge \overline{N}} \rangle$

$\langle elt_{E \wedge \overline{N}} \rangle$ ::= $\langle array_{E \wedge \overline{N}} \rangle$ | $\langle object_{E \wedge \overline{N}} \rangle$

$\langle array_{E \wedge \overline{N}} \rangle$ ::= '[' $\langle elts_{E \wedge \overline{N}} \rangle$ ']'

$\langle object_{E \wedge \overline{N}} \rangle$ ::= '{' $\langle items_{E \wedge \overline{N}} \rangle$ '}'

$\langle elts_{E \wedge \overline{N}} \rangle$ ::= $\langle elt_{E \wedge \overline{N}} \rangle$ | $\langle elt_{E \wedge \overline{N}} \rangle$ ',' $\langle elts_{\overline{N}} \rangle$
| $\langle elt_{\overline{N}} \rangle$ ',' $\langle elts_{E \wedge \overline{N}} \rangle$

$\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$ ::= 'false' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle object_{\overline{N}} \rangle$ | $\langle array_{\overline{N}} \rangle$

$\langle items_{E \wedge \overline{N}} \rangle$ ::= $\langle item_{E \wedge \overline{N}} \rangle$ | $\langle item_{E \wedge \overline{N}} \rangle$ ',' $\langle items_{\overline{N}} \rangle$
| $\langle item_{\overline{N}} \rangle$ ',' $\langle items_{E \wedge \overline{N}} \rangle$

$\langle item_{E \wedge \overline{N}} \rangle$ ::= $\langle string_{E_1} \rangle$ ':' $\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$
| $\langle string \rangle$ ':' $\langle elt_{E \wedge \overline{N} \wedge \overline{N_1}} \rangle$

$\langle elt_{E \wedge \overline{N} \wedge \overline{N_1}} \rangle$ ::= $\langle array_{E \wedge \overline{N}} \rangle$ | $\langle object_{E \wedge \overline{N}} \rangle$

$\langle json_{\overline{N}} \rangle$ ::= $\langle elt_{\overline{N}} \rangle$

$\langle elt_{\overline{N}} \rangle$ ::= 'false' | 'null' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle array_{\overline{N}} \rangle$ | $\langle object_{\overline{N}} \rangle$

$\langle array_{\overline{N}} \rangle$ ::= '[]' | '[' $\langle elts_{\overline{N}} \rangle$ ']'

$\langle object_{\overline{N}} \rangle$ ::= '{}' | '{' $\langle items_{\overline{N}} \rangle$ '}'

$\langle elts_{\overline{N}} \rangle$ ::= $\langle elt_{\overline{N}} \rangle$ | $\langle elt_{\overline{N}} \rangle$ ',' $\langle elts_{\overline{N}} \rangle$

$\langle items_{\overline{N}} \rangle$ ::= $\langle item_{\overline{N}} \rangle$ | $\langle item_{\overline{N}} \rangle$ ',' $\langle items_{\overline{N}} \rangle$

$\langle item_{\overline{N}} \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$

$\langle elt_{\overline{N} \wedge \overline{N_1}} \rangle$ ::= 'false' | 'true'
| $\langle number \rangle$ | $\langle string \rangle$ | $\langle array_{\overline{N}} \rangle$ | $\langle object_{\overline{N}} \rangle$

$\langle json_N \rangle$ ::= $\langle elt_N \rangle$

$\langle elt_N \rangle$ ::= $\langle object_N \rangle$ | $\langle array_N \rangle$

$\langle array_N \rangle$ ::= '[' $\langle elts_N \rangle$ ']'

$\langle object_N \rangle$ ::= '{' $\langle items_N \rangle$ '}'

$\langle elts_N \rangle$ ::= $\langle elt_N \rangle$ ',' $\langle elts \rangle$ | $\langle elt \rangle$ ',' $\langle elts_N \rangle$ | $\langle elt_N \rangle$

$\langle items_N \rangle$ ::= $\langle item_N \rangle$ ',' $\langle items \rangle$ | $\langle item \rangle$ ',' $\langle items_N \rangle$
| $\langle item\ N \rangle$

$\langle item_N \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_N \rangle$ | $\langle string \rangle$ ':' $\langle elt_{N_1} \rangle$

$\langle elt_{N_1} \rangle$ ::= 'null'

$\langle json_E \rangle$ ::= $\langle elt_E \rangle$

$\langle elt_E \rangle$ ::= $\langle object_E \rangle$ | $\langle array_E \rangle$

$\langle object_E \rangle$ ::= '{' $\langle items_E \rangle$ '}'

$\langle items_E \rangle$ ::= $\langle item_E \rangle$ ',' $\langle items \rangle$ | $\langle item \rangle$ ',' $\langle items_E \rangle$
| $\langle item_E \rangle$

$\langle item_E \rangle$ ::= $\langle string \rangle$ ':' $\langle elt_E \rangle$

$\langle array_E \rangle$ ::= '[' $\langle elts_E \rangle$ ']'

$\langle elts_E \rangle$ ::= $\langle elt_E \rangle$ ',' $\langle elts \rangle$ | $\langle elt \rangle$ ',' $\langle elts_E \rangle$ | $\langle elt_E \rangle$

Supercharged Pattern Matchers

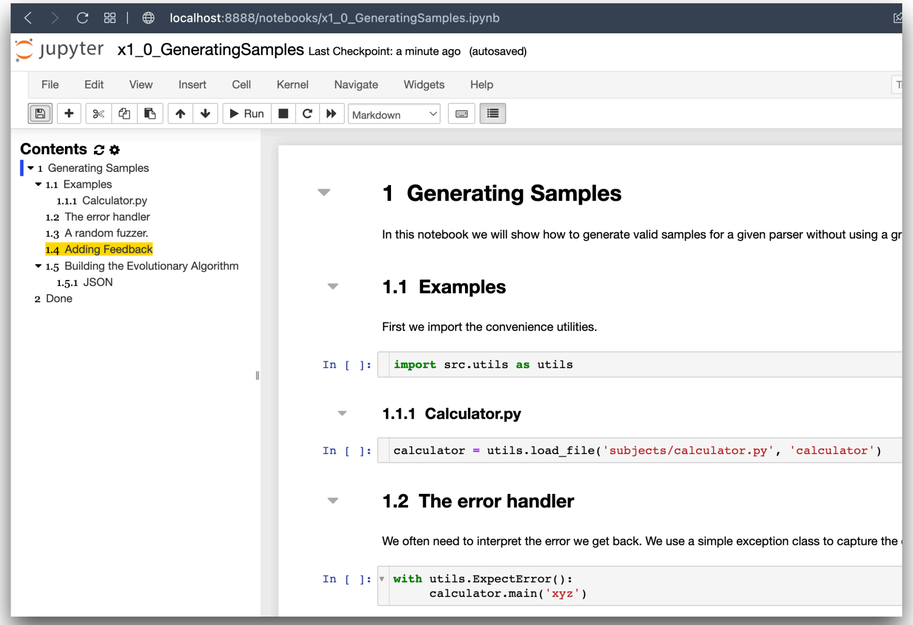
```
<calc not(D or F)>
  where
    <factor F> is ((<expr>))
    <term D> is <factor> / 0
```

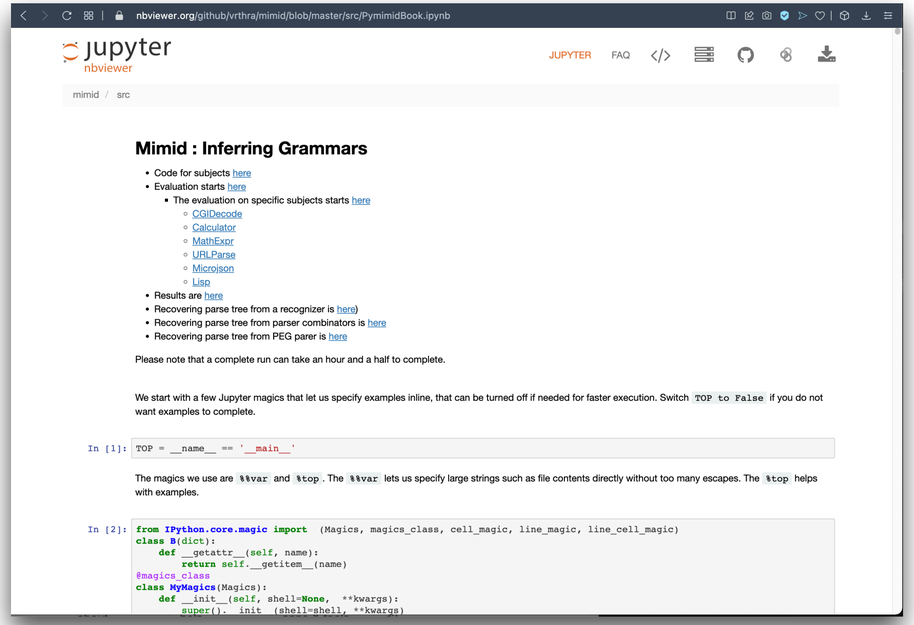
```
<json E and not(N)>
  where
    <item E> is "": <elt>
    <item N> is <string>:null
```

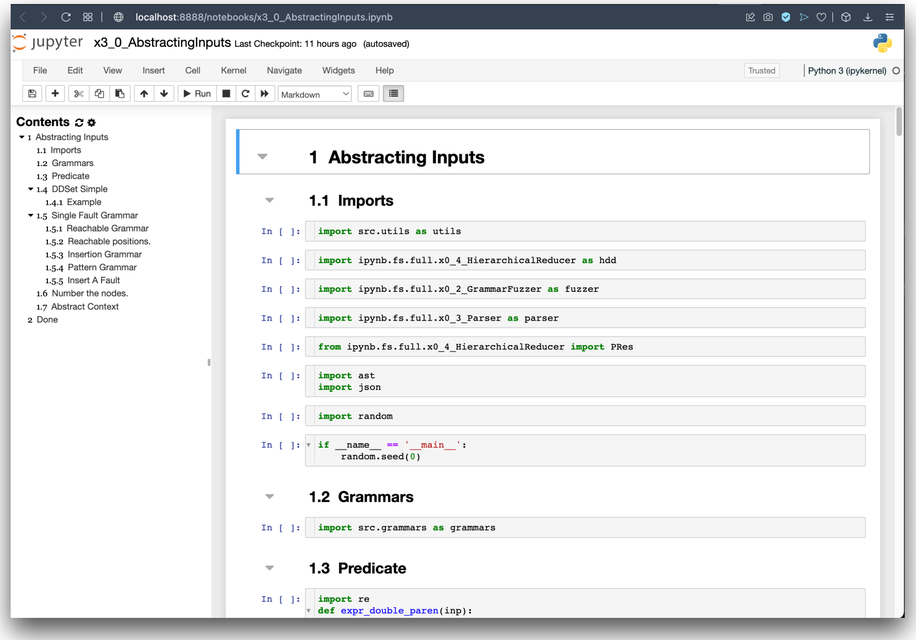
```
<C not(F) not(EW or ED or F)>
  where
    <forCondition F> is ";;"
    <iterationStatement EW> is <WHILE> "()" <statement>
    <iterationStatement ED> is <DO> <statement> <WHILE> "()" <eos>
```

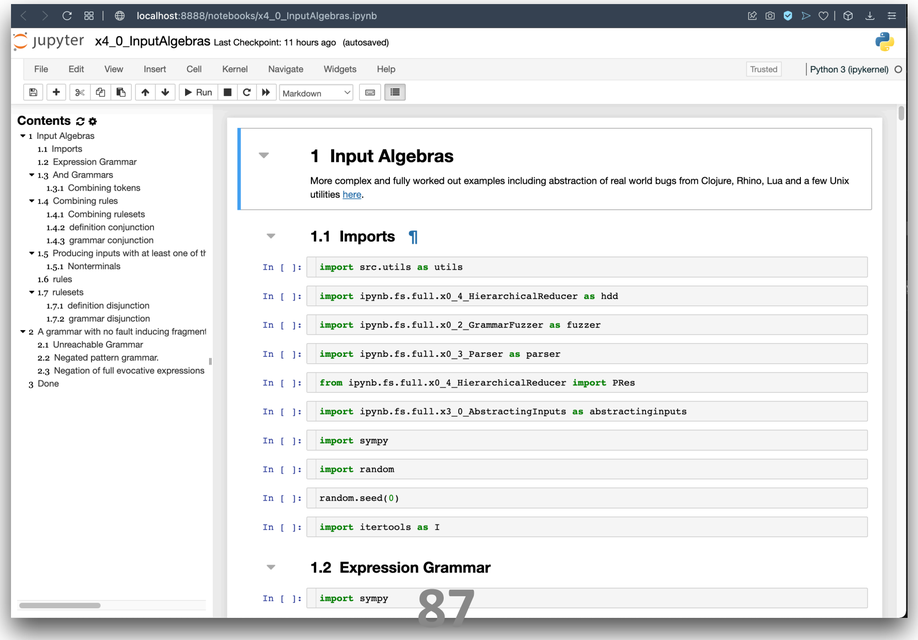
```
<ipv4addr O and H>
  where
    <quad O> is "0" <num>
    <quad H> is "0x" <num>
```

Alternative to Regular Expressions


localhost:8888/notebooks/x1_0_GeneratingSamples.ipynb
jupyter x1_0_GeneratingSamples Last Checkpoint: a minute ago (autosaved)
File Edit View Insert Cell Kernel Navigate Widgets Help
Run Markdown
Contents
1 Generating Samples
1.1 Examples
1.1.1 Calculator.py
1.2 The error handler
1.3 A random fuzzer.
1.4 Adding Feedback
1.5 Building the Evolutionary Algorithm
1.5.1 JSON
2 Done
1 Generating Samples
In this notebook we will show how to generate valid samples for a given parser without using a gr
1.1 Examples
First we import the convenience utilities.
In [ ]: import src.utils as utils
1.1.1 Calculator.py
In [ ]: calculator = utils.load_file('subjects/calculator.py', 'calculator')
1.2 The error handler
We often need to interpret the error we get back. We use a simple exception class to capture the
In [ ]: with utils.ExpectError():
calculator.main('xyz')


nbviewer.org/github/vrthra/mimid/blob/master/src/PymimidBook.ipynb
jupyter nbviewer
JUPYTER FAQ
mimid / src
Mimid : Inferring Grammars
Code for subjects here
Evaluation starts here
The evaluation on specific subjects starts here
CGIDecode
Calculator
MathExpr
URLParse
Microjson
Lisp
Results are here
Recovering parse tree from a recognizer is here)
Recovering parse tree from parser combinators is here
Recovering parse tree from PEG parer is here
Please note that a complete run can take an hour and a half to complete.
We start with a few Jupyter magics that let us specify examples inline, that can be turned off if needed for faster execution. Switch TOP to False if you do not want examples to complete.
In [1]: TOP = __name__ == '__main__'
The magics we use are %%var and %top. The %%var lets us specify large strings such as file contents directly without too many escapes. The %top helps with examples.
In [2]: from IPython.core.magic import (Magics, magics_class, cell_magic, line_magic, line_cell_magic)
class B(dict):
def __getattr__(self, name):
return self.__getitem__(name)
@magics_class
class MyMagics(Magics):
def __init__(self, shell=None, **kwargs):


localhost:8888/notebooks/x3_0_AbstractingInputs.ipynb
jupyter x3_0_AbstractingInputs Last Checkpoint: 11 hours ago (autosaved)
File Edit View Insert Cell Kernel Navigate Widgets Help
Trusted Python 3 (ipykernel)
Run Markdown
Contents
1 Abstracting Inputs
1.1 Imports
1.2 Grammars
1.3 Predicate
1.4 DDSet Simple
1.4.1 Example
1.5 Single Fault Grammar
1.5.1 Reachable Grammar
1.5.2 Reachable positions.
1.5.3 Insertion Grammar
1.5.4 Pattern Grammar
1.5.5 Insert A Fault
1.6 Number the nodes.
1.7 Abstract Context
2 Done
1 Abstracting Inputs
1.1 Imports
In [ ]: import src.utils as utils
In [ ]: import ipynb.fs.full.x0_4_HierarchicalReducer as hdd
In [ ]: import ipynb.fs.full.x0_2_GrammarFuzzer as fuzzer
In [ ]: import ipynb.fs.full.x0_3_Parser as parser
In [ ]: from ipynb.fs.full.x0_4_HierarchicalReducer import PRes
In [ ]: import ast
import json
In [ ]: import random
In [ ]: if __name__ == '__main__':
random.seed(0)
1.2 Grammars
In [ ]: import src.grammars as grammars
1.3 Predicate
In [ ]: import re
def expr_double_paren(inp):


localhost:8888/notebooks/x4_0_InputAlgebras.ipynb
jupyter x4_0_InputAlgebras Last Checkpoint: 11 hours ago (autosaved)
File Edit View Insert Cell Kernel Navigate Widgets Help
Trusted Python 3 (ipykernel)
Run Markdown
Contents
1 Input Algebras
1.1 Imports
1.2 Expression Grammar
1.3 And Grammars
1.3.1 Combining tokens
1.4 Combining rules
1.4.1 Combining rulesets
1.4.2 definition conjunction
1.4.3 grammar conjunction
1.5 Producing inputs with at least one of t
1.5.1 Nonterminals
1.6 rules
1.7 rulesets
1.7.1 definition disjunction
1.7.2 grammar disjunction
2 A grammar with no fault inducing fragment
2.1 Unreachable Grammar
2.2 Negated pattern grammar.
2.3 Negation of full evocative expressions
3 Done
1 Input Algebras
More complex and fully worked out examples including abstraction of real world bugs from Clojure, Rhino, Lua and a few Unix utilities here.
1.1 Imports ¶
In [ ]: import src.utils as utils
In [ ]: import ipynb.fs.full.x0_4_HierarchicalReducer as hdd
In [ ]: import ipynb.fs.full.x0_2_GrammarFuzzer as fuzzer
In [ ]: import ipynb.fs.full.x0_3_Parser as parser
In [ ]: from ipynb.fs.full.x0_4_HierarchicalReducer import PRes
In [ ]: import ipynb.fs.full.x3_0_AbstractingInputs as abstractinginputs
In [ ]: import sympy
In [ ]: import random
In [ ]: random.seed(0)
In [ ]: import itertools as I
1.2 Expression Grammar
In [ ]: import sympy