Cartesian configuration format file parser.

Filter syntax:

- `,` means `OR`
- `..` means `AND`
- `.` means `IMMEDIATELY-FOLLOWED-BY`
- `(xx=yy)` where `xx=VARIANT_NAME` and `yy=VARIANT_VALUE`

Example:

```
qcow2..(guest_os=Fedora).14, RHEL.6..raw..boot, smp2..qcow2..migrate..ide
```

means match all dicts whose names have:

```
(qcow2 AND ((guest_os=Fedora) IMMEDIATELY-FOLLOWED-BY 14)) OR
((RHEL IMMEDIATELY-FOLLOWED-BY 6) AND raw AND boot) OR
(smp2 AND qcow2 AND migrate AND ide)
```

Note:

- `qcow2..Fedora.14` is equivalent to `Fedora.14..qcow2`.
- `qcow2..Fedora.14` is not equivalent to `qcow2..14.Fedora`.
- `ide, scsi` is equivalent to `scsi, ide`.

Filters can be used in 3 ways:

```
only <filter>
no <filter>
<filter>:
```

The last one starts a conditional block.

Formal definition: Regexp come from python. They're not deterministic, but more readable for people. Spaces between terminals and nonterminals are only for better reading of definitions.

The base of the definitions come verbatim as follows:

```
E = {\\n, #, :, "-", =, +=, <=, ~=, ?=, ?+=, ?<=, !, < , del, @, variants, include, only, no, name, value}

N = {S, DEL, FILTER, FILTER_NAME, FILTER_GROUP, PN_FILTER_GROUP, STAT, VARIANT, VAR-TYPE, VAR-NAME, VAR-NAME-F, VAR, COMMENT, TEXT, DEPS, DEPS-NAME-F, META-DATA, IDENTIFIER}``


I = I^n | n in N             // indentation from start of line
                             // where n is indentation length.
I = I^n+x | n,x in N         // indentation with shift

start symbol = S
end symbol = eps

S -> I^0+x STATV | eps

I^n    STATV
I^n    STATV

I^n STATV -> I^n STATV \\n I^n STATV | I^n STAT | I^n variants VARIANT
```

```
I^n STAT -> I^n STAT \\n I^n STAT | I^n COMMENT | I^n include INC
I^n STAT -> I^n del DEL | I^n FILTER

DEL -> name \\n

I^n STAT -> I^n name = VALUE | I^n name += VALUE | I^n name <= VALUE | I^n name ~= VALUE
I^n STAT -> I^n name ?= VALUE | I^n name ?+= VALUE | I^n name ?<= VALUE

VALUE -> TEXT \\n | 'TEXT' \\n | "TEXT" \\n
```

```
COMMENT_BLOCK -> #TEXT | //TEXT
COMMENT ->  COMMENT_BLOCK\\n
COMMENT ->  COMMENT_BLOCK\\n
```

```
TEXT = [^\\n] TEXT            //python format regexp

I^n    variants VAR #comments:             add possibility for comment
I^n+x       VAR-NAME: DEPS
I^n+x+x2        STATV
I^n         VAR-NAME:

IDENTIFIER -> [A-Za-z0-9][A-Za-z0-9_-]*

VARIANT -> VAR COMMENT_BLOCK\\n I^n+x VAR-NAME
VAR -> VAR-TYPE: | VAR-TYPE META-DATA: | :         // Named | unnamed variant

VAR-TYPE -> IDENTIFIER

variants _name_ [xxx] [zzz=yyy] [uuu]:

META-DATA -> [IDENTIFIER] | [IDENTIFIER=TEXT] | META-DATA META-DATA

I^n VAR-NAME -> I^n VAR-NAME \\n I^n VAR-NAME | I^n VAR-NAME-N \\n I^n+x STATV
VAR-NAME-N -> - @VAR-NAME-F: DEPS | - VAR-NAME-F: DEPS
VAR-NAME-F -> [a-zA-Z0-9\\._-]+                 // Python regexp

DEPS -> DEPS-NAME-F | DEPS-NAME-F,DEPS
DEPS-NAME-F -> [a-zA-Z0-9\\._- ]+               // Python regexp

INC -> name \\n


FILTER_GROUP: STAT
    STAT

I^n STAT -> I^n PN_FILTER_GROUP | I^n ! PN_FILTER_GROUP

PN_FILTER_GROUP -> FILTER_GROUP: \\n I^n+x STAT
PN_FILTER_GROUP -> FILTER_GROUP: STAT \\n I^n+x STAT

only FILTER_GROUP
no FILTER_GROUP

FILTER -> only FILTER_GROUP \\n | no FILTER_GROUP \\n

FILTER_GROUP -> FILTER_NAME
FILTER_GROUP -> FILTER_GROUP..FILTER_GROUP
FILTER_GROUP -> FILTER_GROUP,FILTER_GROUP

FILTER_NAME -> FILTER_NAME.FILTER_NAME
FILTER_NAME -> VAR-NAME-F | (VAR-NAME-F=VAR-NAME-F)
```