

Nextflow cheatsheet: creating input channels

Create a channel

Channel.of("A")

• Channel.of() or Channel.fromList() will not alter anything

Channel.fromPath("A")

Channel.fromPath("/path/A")

• If the full file path is absent, .fromPath() will prefix current folder as path
• So the resulting channels always carry a full path

Channel content

A

/path/A

Use the channel in the process

input: val(x) "echo \$x"

input: path(x) "cat \$x"

or input: path("x.txt") "cat x.txt"

• input: path("x.txt") will create a **symlink** to /path/A in the working directory with the name "x.txt"

• Creating input channels from a text file

Input text file content

A
B
C

Create a channel

**Channel.fromPath("file.txt")
.splitText { it.strip() }**

• .splitText will read in the file.txt
• it.strip() will remove the blank items

**Channel.fromPath("file.txt")
.splitText { it.strip() }
.map { it -> file(it) }**

• file() adds current folder as path unless there is already full path in the item

Channel content

A
B
C

/path/A
/path/B
/path/C

• 1 channel, 3 items
• 3 parallel executions of process

• 1 channel, 3 items
• 3 parallel executions of process

Use the channel in the process

input: val(x)

input: path(x)

A	/path/A.bam	/path/A.bam.bai
B	/path/B.bam	/path/B.bam.bai
C	/path/C.bam	/path/C.bam.bai

Channel.fromPath("file.tsv")

**.splitCsv(sep: "\t")
.map { row -> [row[0], file(row[1]), file(row[2])] }**

• .map{ } is very useful to select columns and specify channel structure. Here it converts the row to a tuple

[A, /path/A.bam, /path/A.bam.bai]
[B, /path/B.bam, /path/B.bam.bai]
[C, /path/C.bam, /path/C.bam.bai]

input: tuple val(x), path(bam), path(bai)

strain	bam
A	A.bam
B	B.bam
C	C.bam

Channel.fromPath("file.tsv")

**.splitCsv(header:true, sep: "\t")
.map { row ->
if (params.bam_path != "") {
row.bam = "\${params.bam_path}/\${row.bam}"
}
[row.strain, file("\${row.bam}"), file("\${row.bam}.bai")] }**

• A path is added with params.bam_path
• If "params.bam_path" doesn't add a full path, file() will.
• [x, y] is the same as tuple(x, y)

• Creating input channels from a list of files

Files in the folder

A_R1.fq
A_R2.fq
B_R1.fq
B_R2.fq

A.bam
A.bam.bai
B.bam
B.bam.bai

Create a channel

Channel.fromPath("*.fq")

Channel.fromFilePairs("/path/*_R{1,2}.fq", flat:true)

Channel.fromFilePairs("/path/*.{bam, bam.bai}", flat:true)

Channel content

/path/A_R1.fq
/path/A_R2.fq
/path/B_R1.fq
/path/B_R2.fq

Use the channel in the process

input: path(fq)
• 1 channel, 4 items,
• 4 parallel executions of process

[A, /path/A_R1.fq, /path/A_R2.fq]
[B, /path/B_R1.fq, /path/B_R2.fq]

input: tuple val(x), path(R1), path(R2)

input: tuple val(x), path(bam), path(bai)

• First item "A" came from stripping the path and common pattern ".{bam, bam.bai}" as specified in .Channel.fromFilePairs

Nextflow cheatsheet: combine inputs into 1 channel

examples of 1 channel:

`Channel.of("A.fa", "B.fa", "C.fa")`

A.fa →
B.fa →
C.fa →

- Each item (row) has 1 execution of the process
- Here is 1 channel with 3 items
- 3 parallel executions of the process

`Channel.of(["A.fa", "B.fa", "C.fa"])`

[A.fa, B.fa, C.fa] →

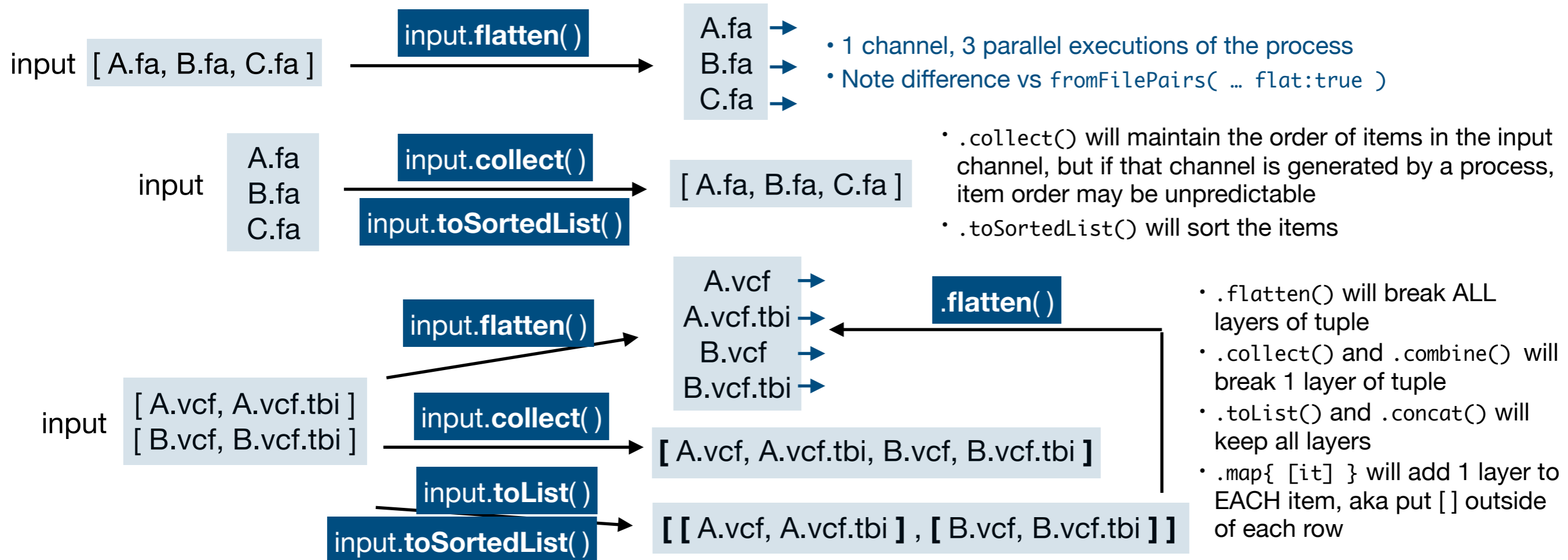
- 1 channel, 1 item, 1 execution

`Channel.of(["I.vcf", "I.vcf.tbi"],
["II.vcf", "II.vcf.tbi"],
["III.vcf", "III.vcf.tbi"])`

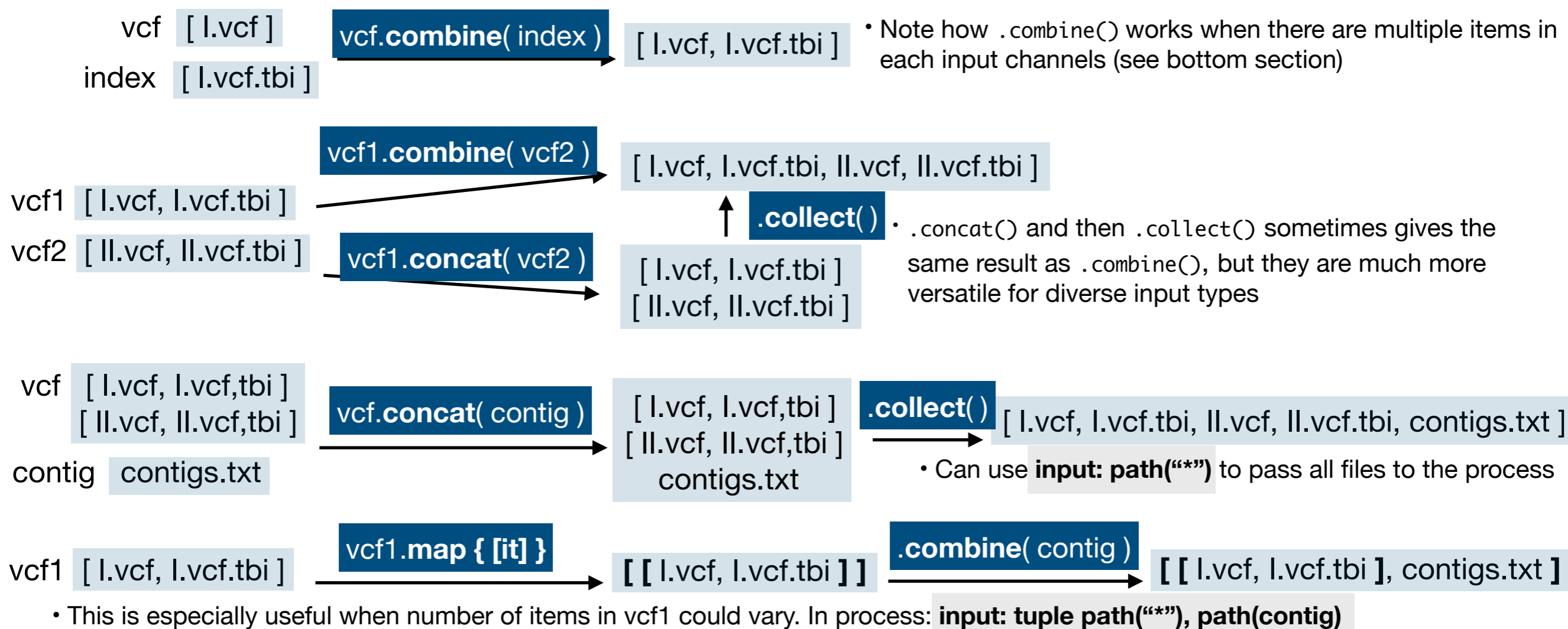
[I.vcf, I.vcf.tbi] →
[II.vcf, II.vcf.tbi] →
[III.vcf, III.vcf.tbi] →

- 1 channel, 3 items
- 3 parallel executions of process
- [] indicates a "set" "tuple" "ArrayList"

• With 1 input channel: change number of items and parallel execution of the process



• With multiple input channels: combine them and change number of items



• Other useful cases

