

TopHat: a stylish journey through modular interactive workflows

Tim Steenvoorden

Markus Klinik

Nico Naus

Dutch FP Day, January 11, 2019



TopHat: a stylish journey through modular interactive workflows

Tim Steenvoorden

Markus Klinik

Nico Naus

Dutch FP Day, January 11, 2019



Task Oriented Programming (TOP)

Workflows

coordinate collaboration

Interactive

driven by user input

Modular

higher order

Task Oriented Programming (TOP)

Workflows

coordinate collaboration

Interactive

driven by user input

Modular

higher order

elementary building blocks and concepts



Task Oriented Programming (TOP)

Workflows

coordinate collaboration

Interactive

driven by user input

Modular

higher order

elementary building blocks and concepts

labeled transition system



Task Oriented Programming (TOP)

Workflows

coordinate collaboration

Interactive

driven by user input

Modular

higher order

elementary building blocks and concepts

labeled transition system

embedding in simply typed λ -calculus
(with clearly separated semantics)



Task Oriented Programming (TOP)

Workflows

coordinate collaboration

Interactive

driven by user input

Modular

higher order

elementary building blocks and concepts

labeled transition system

embedding in simply typed λ -calculus
(with clearly separated semantics)



Foundation for formal reasoning
and comparison to other frameworks

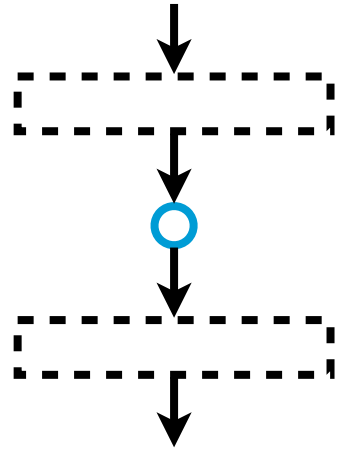


In perspective

Collaboration

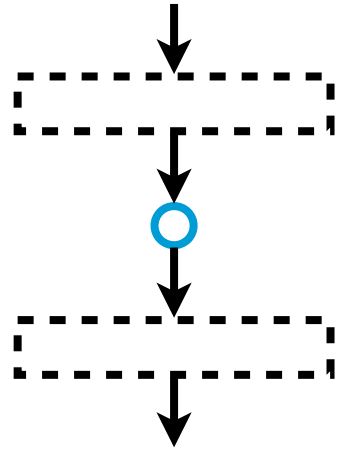


Collaboration

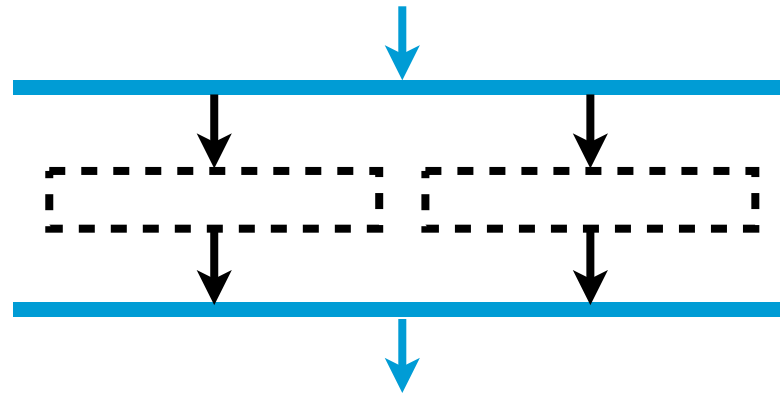


AFTER EACH OTHER

Collaboration

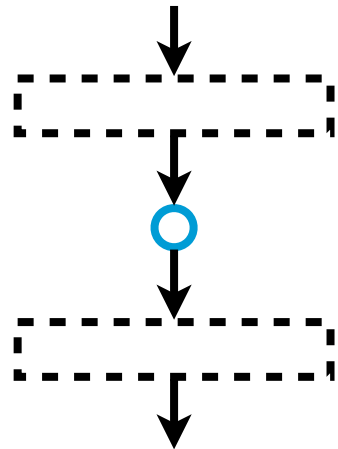


AFTER EACH OTHER

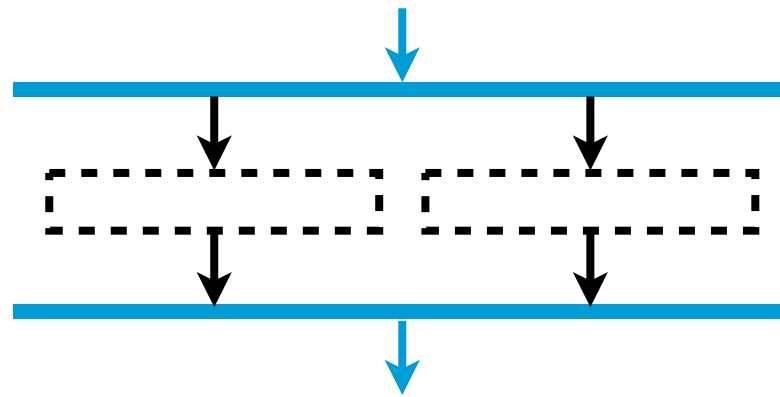


AT THE SAME TIME

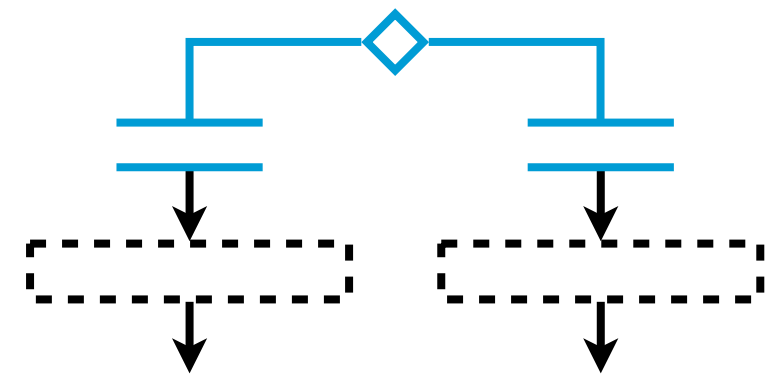
Collaboration



AFTER EACH OTHER

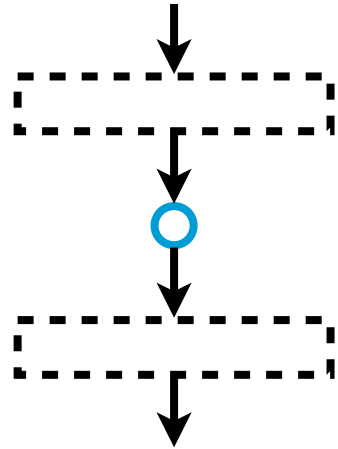


AT THE SAME TIME

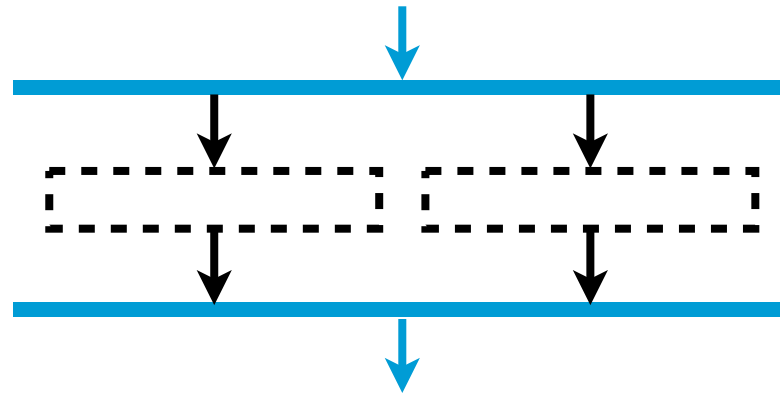


CONDITIONALLY

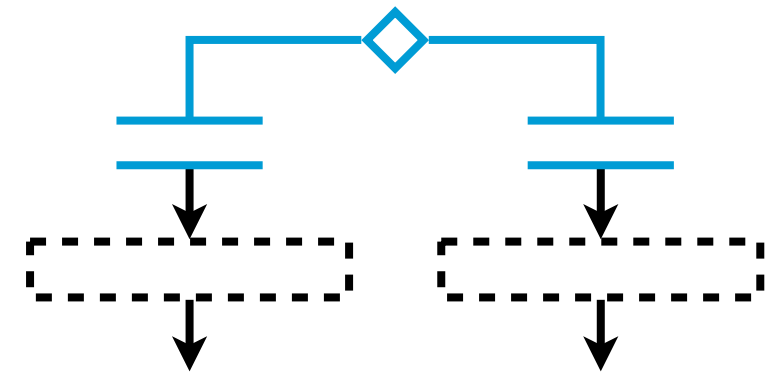
Collaboration



AFTER EACH OTHER



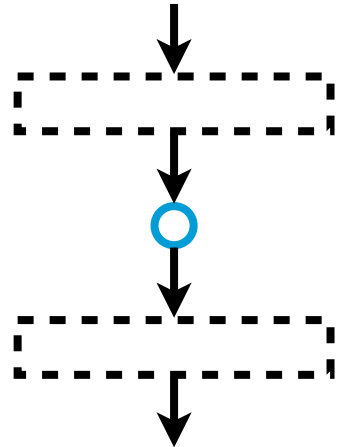
AT THE SAME TIME



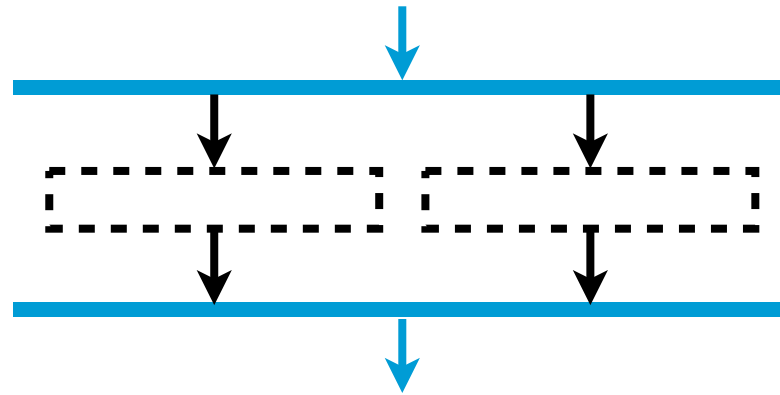
CONDITIONALLY

Communication is taken care of!

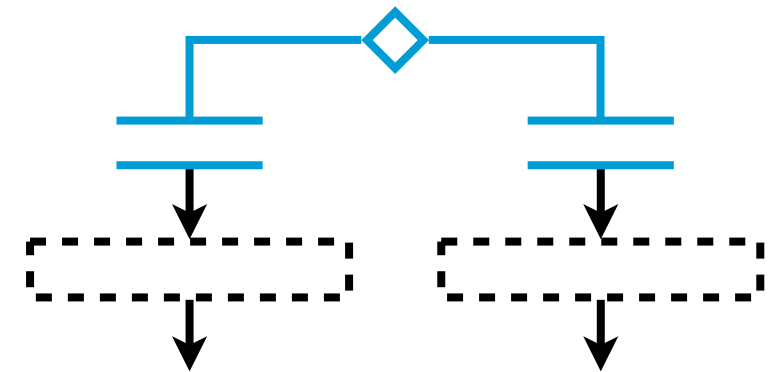
Collaboration



AFTER EACH OTHER



AT THE SAME TIME



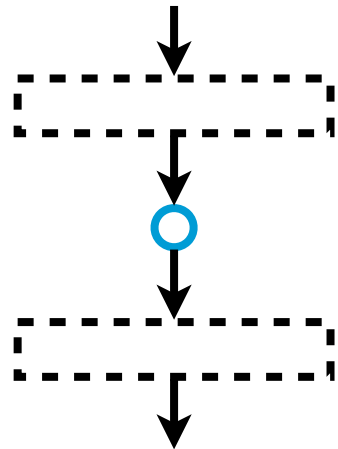
CONDITIONALLY

Communication is taken care of!

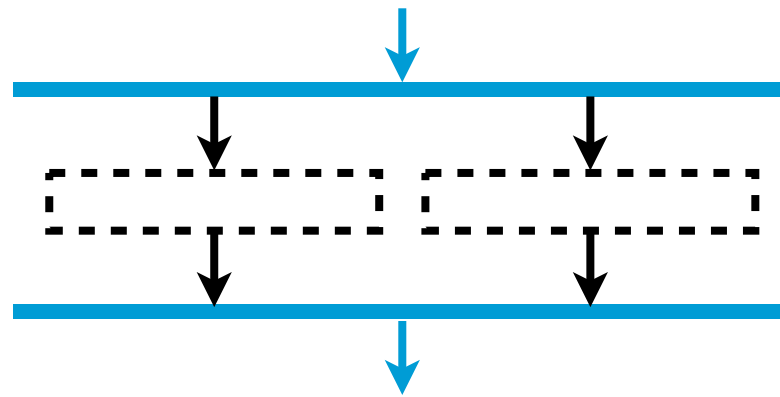
“We do A and B at the same time, then we continue with C .”



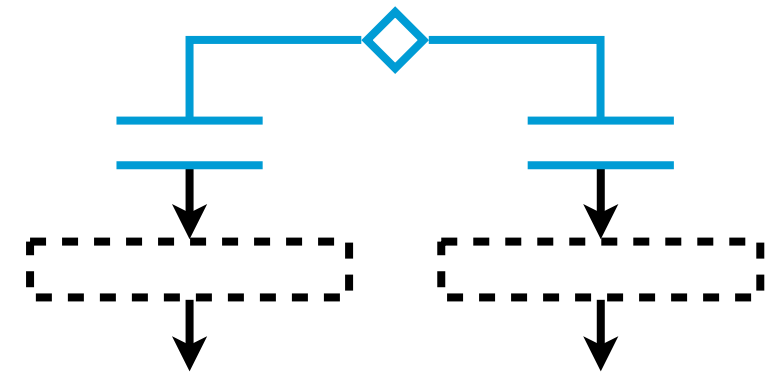
Collaboration



AFTER EACH OTHER



AT THE SAME TIME



CONDITIONALLY

Communication is taken care of!

“We do A and B at the same time, then we continue with C .”



“When I send you M , you can do B and I’ll wait for you to send N .”

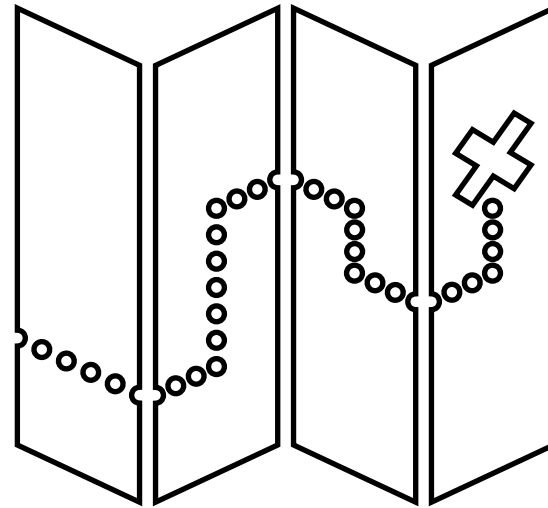




Building blocks

Editors

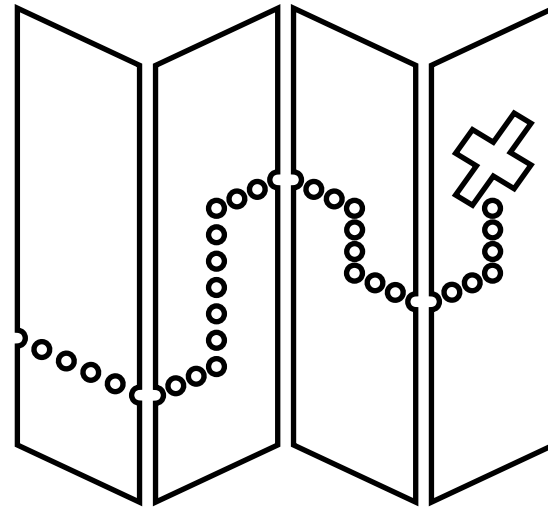
5



Editors

5

□5

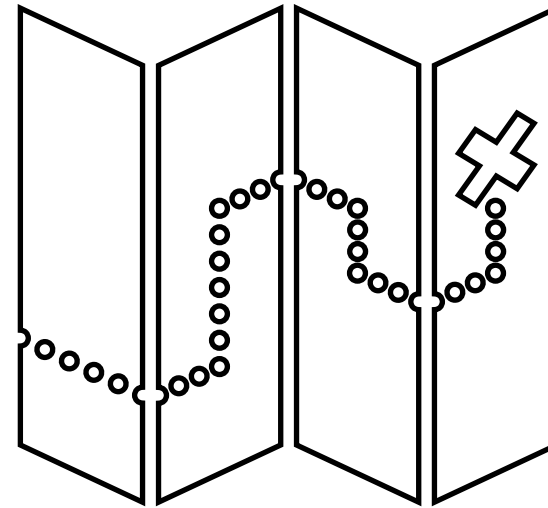
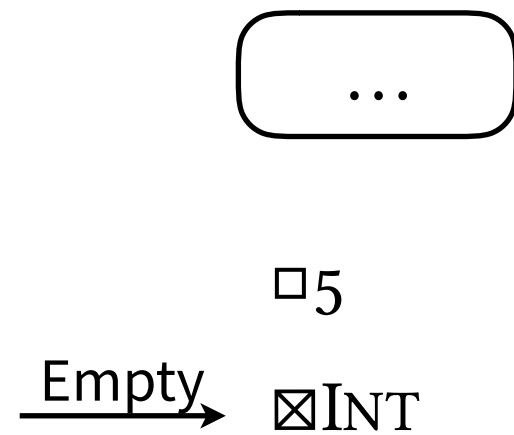


□(51°49'N, 5°52'E)



□True

Editors

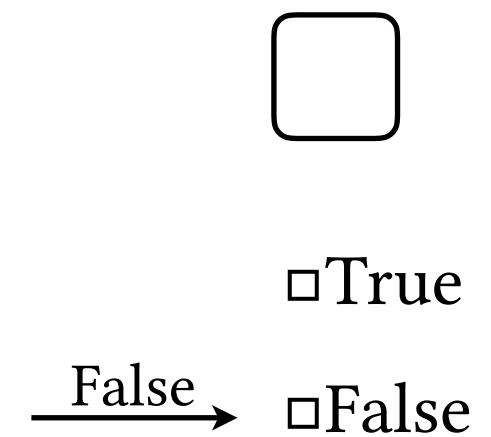
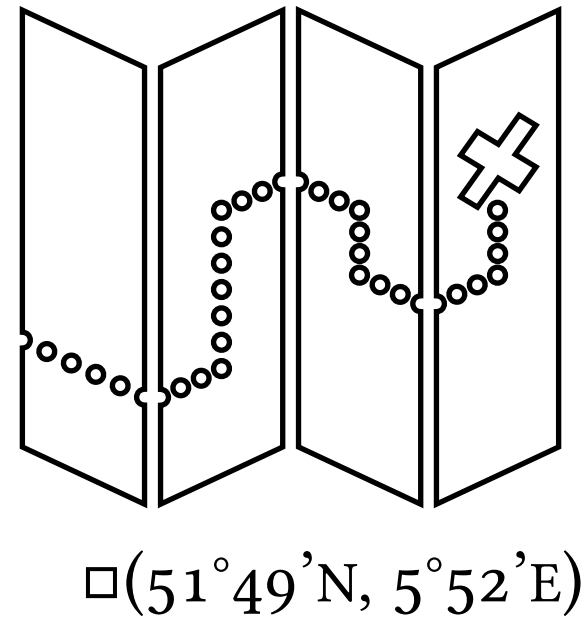
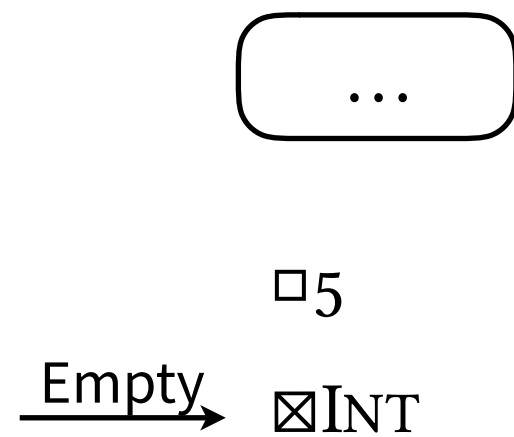


□(51°49'N, 5°52'E)

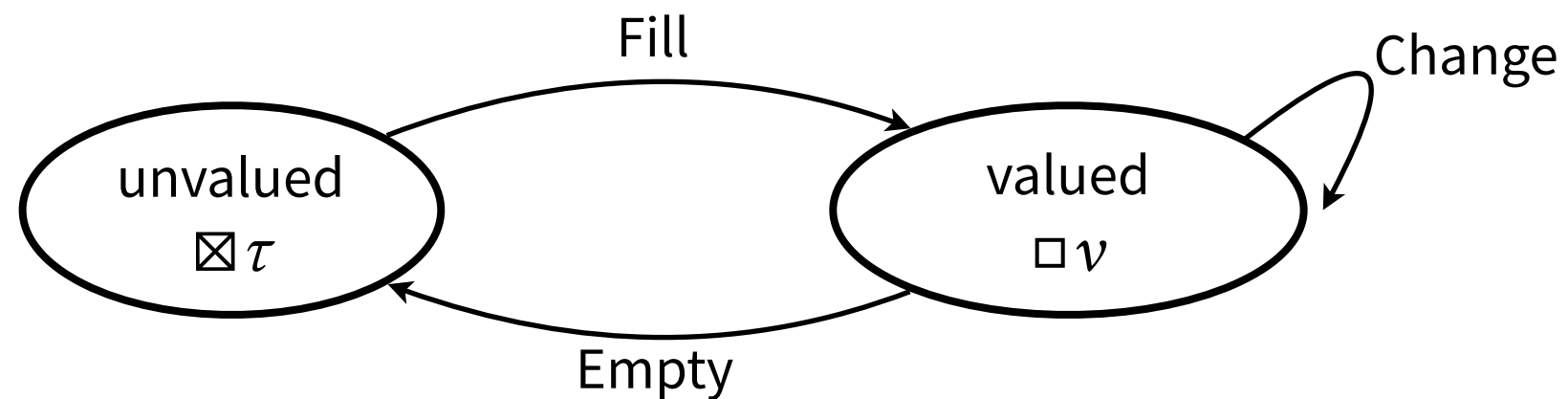
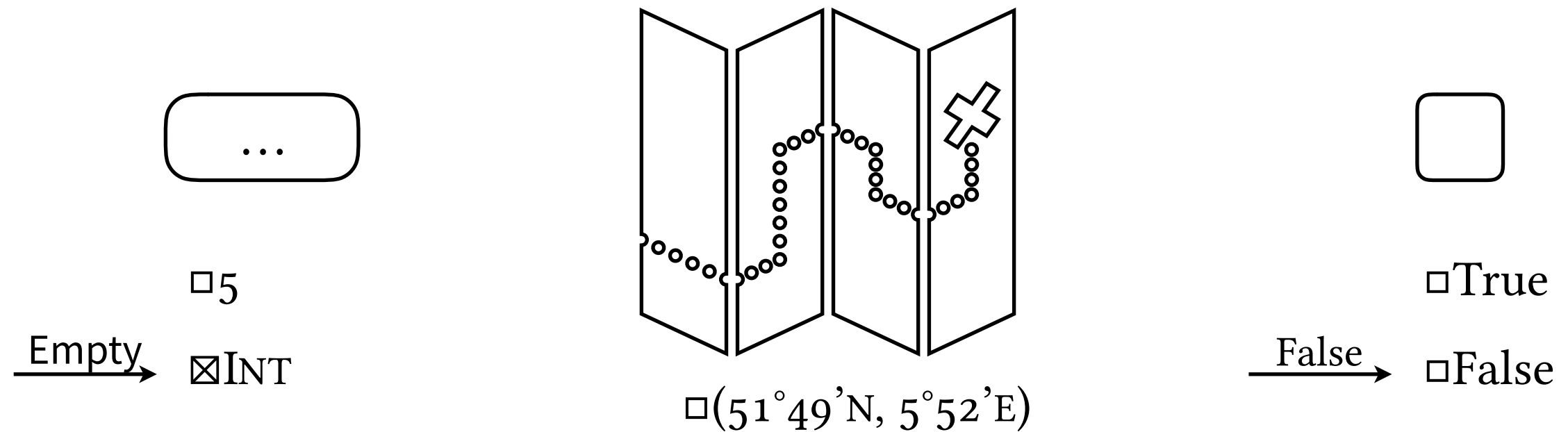


□True

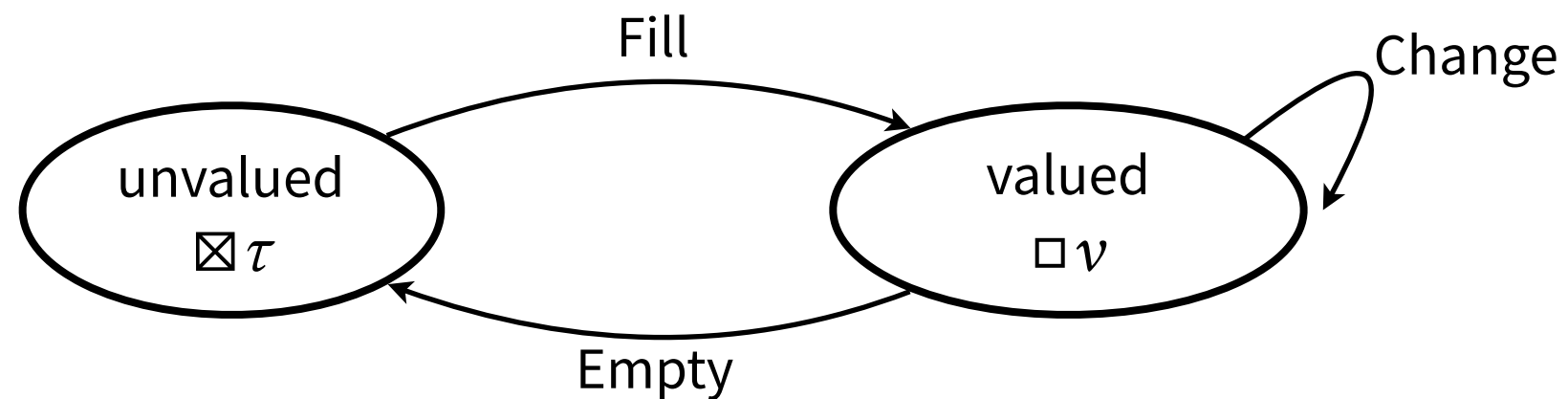
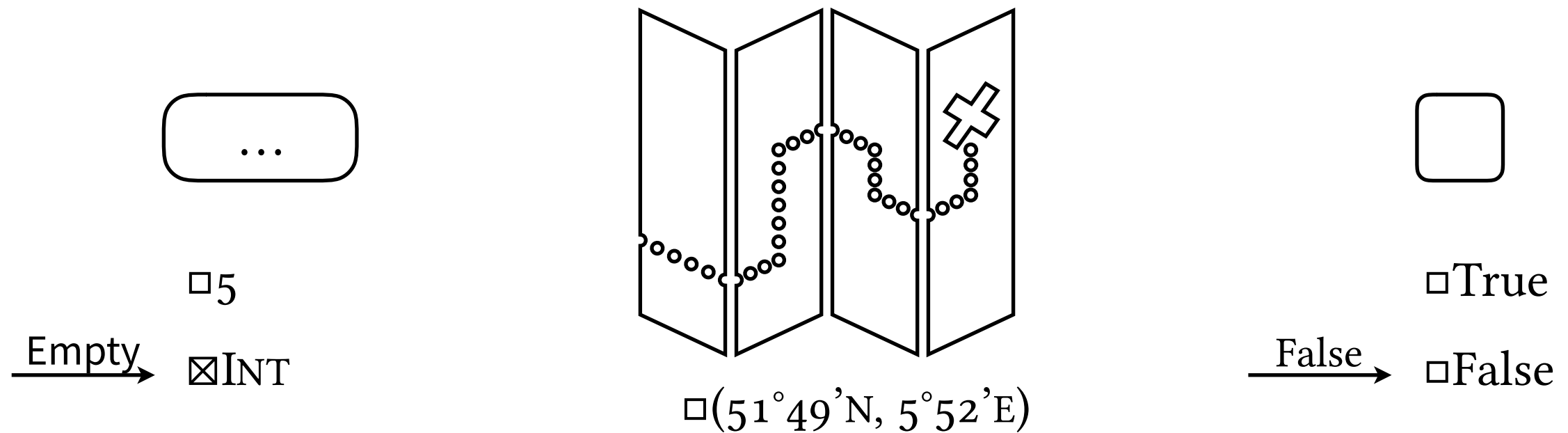
Editors



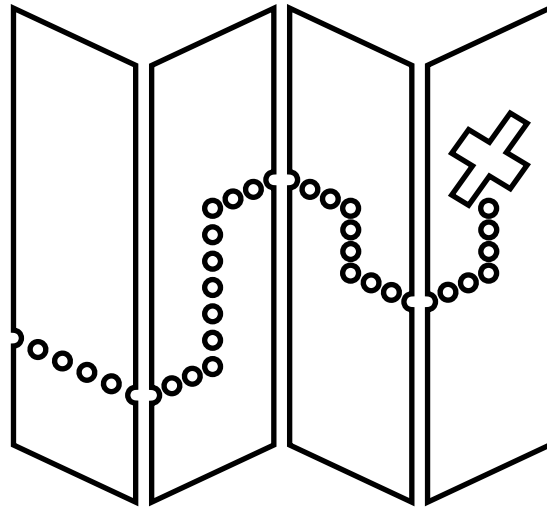
Editors



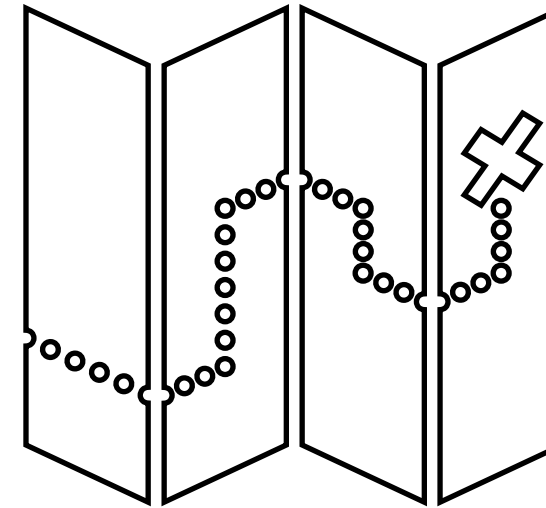
Editors



Shared editors

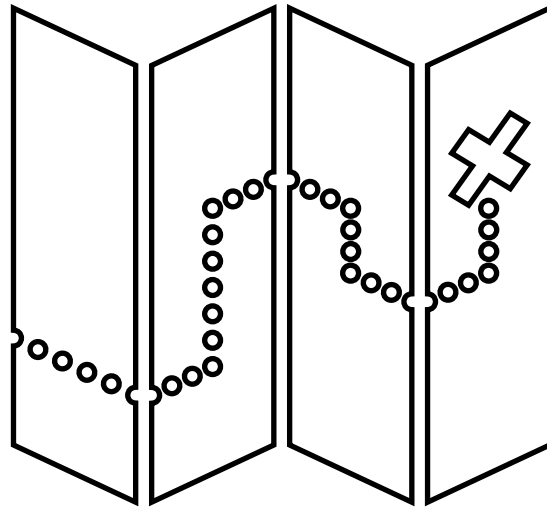


■(51°49'N, 5°52'E)

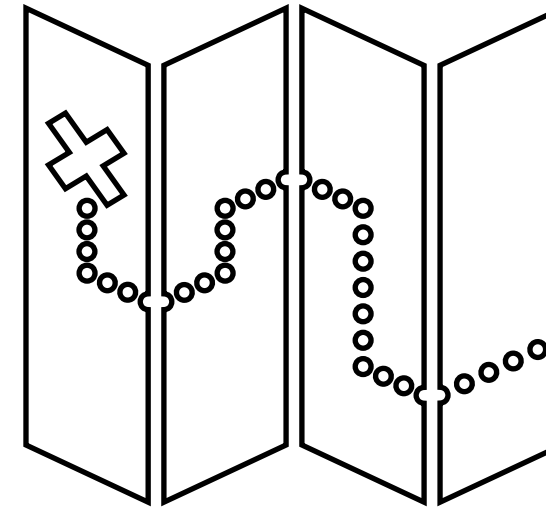


■(51°49'N, 5°52'E)

Shared editors



■ $(51^{\circ}49'N, 5^{\circ}52'E)$

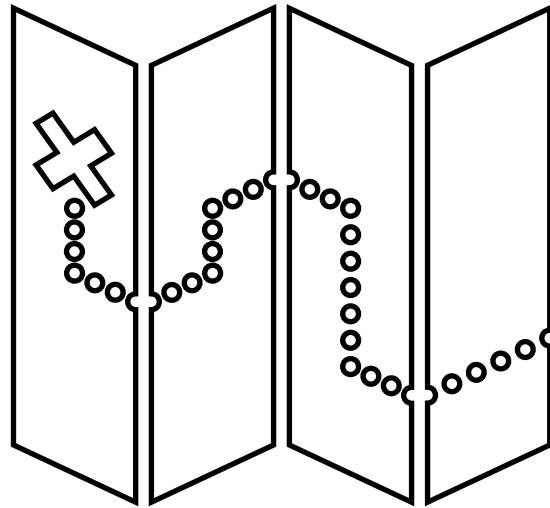


■ $(51^{\circ}49'N, 5^{\circ}52'E)$

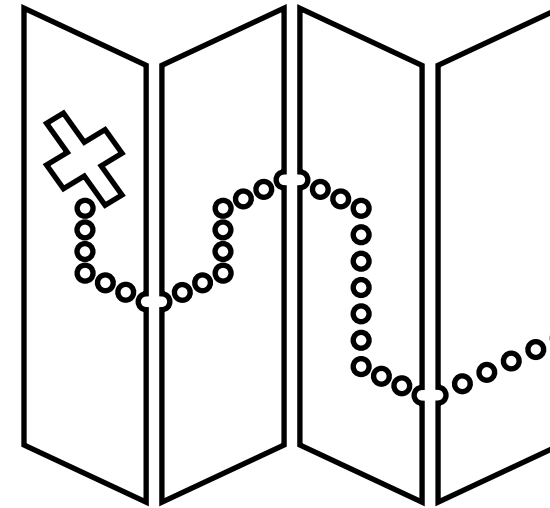
$(51^{\circ}49'N, 5^{\circ}52'W)$
→

■ $(51^{\circ}49'N, 5^{\circ}52'W)$

Shared editors



■ $(51^{\circ}49'N, 5^{\circ}52'E)$

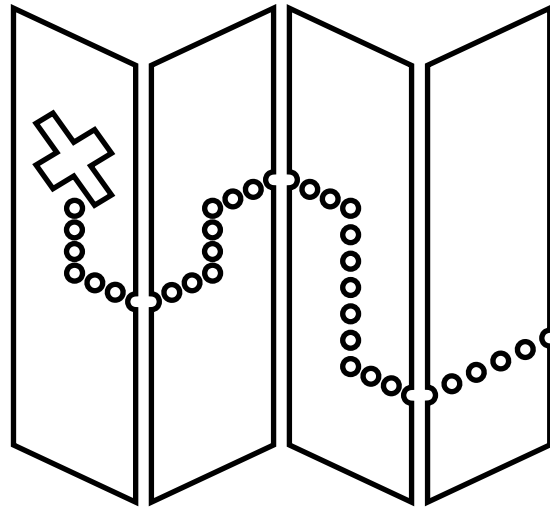


■ $(51^{\circ}49'N, 5^{\circ}52'E)$

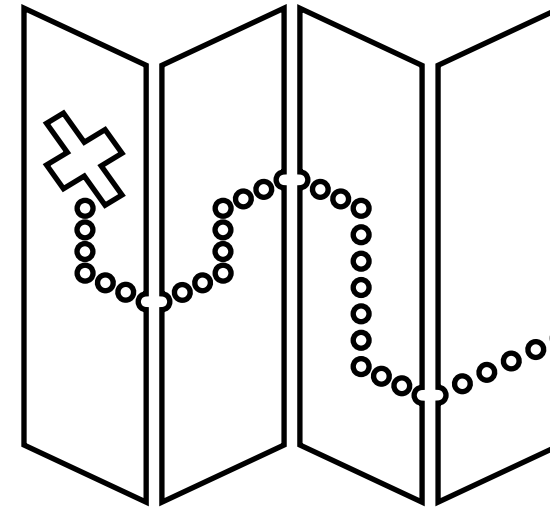
$(51^{\circ}49'N, 5^{\circ}52'W)$
→

■ $(51^{\circ}49'N, 5^{\circ}52'W)$

Shared editors



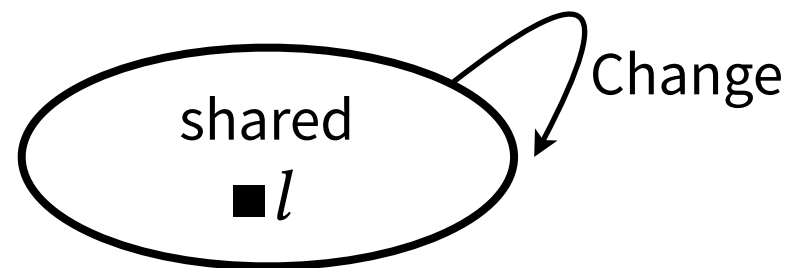
■(51°49'N, 5°52'E)



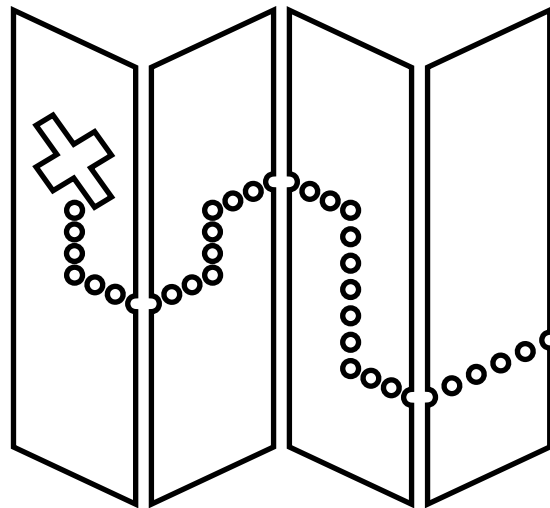
■(51°49'N, 5°52'E)

(51°49'N, 5°52'W)

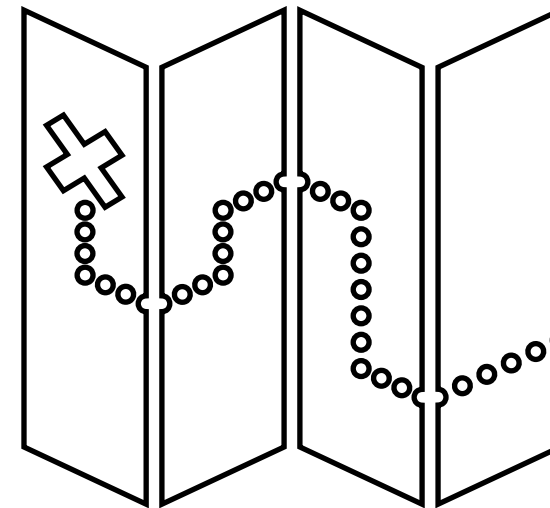
■(51°49'N, 5°52'W)



Shared editors



■(51°49'N, 5°52'E)



■(51°49'N, 5°52'E)

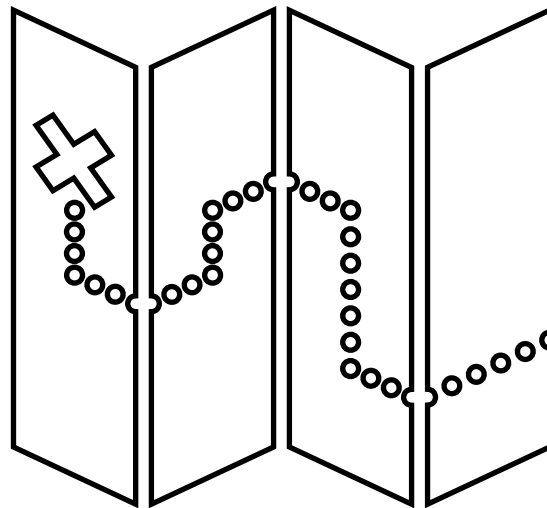
(51°49'N, 5°52'W)

■(51°49'N, 5°52'W)

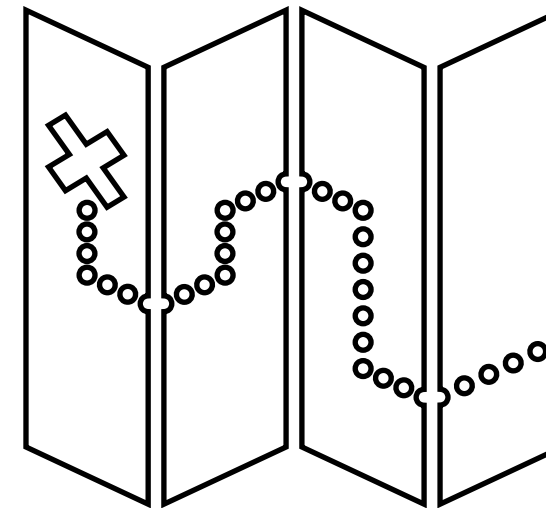


WATCHES A REFERENCE

Shared editors



■(51°49'N, 5°52'E)



■(51°49'N, 5°52'E)

(51°49'N, 5°52'W)

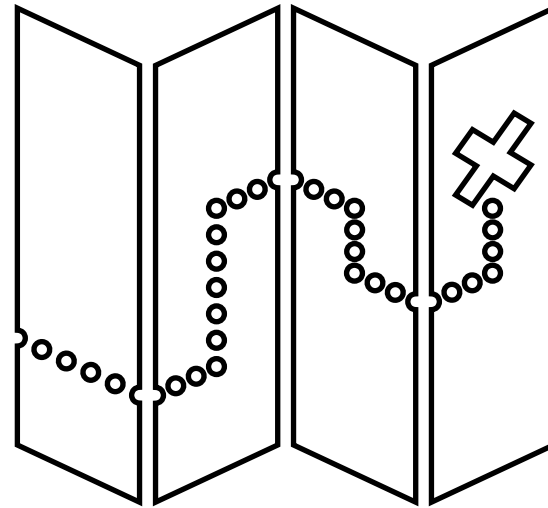
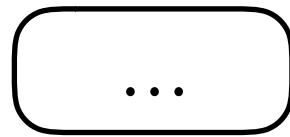
■(51°49'N, 5°52'W)



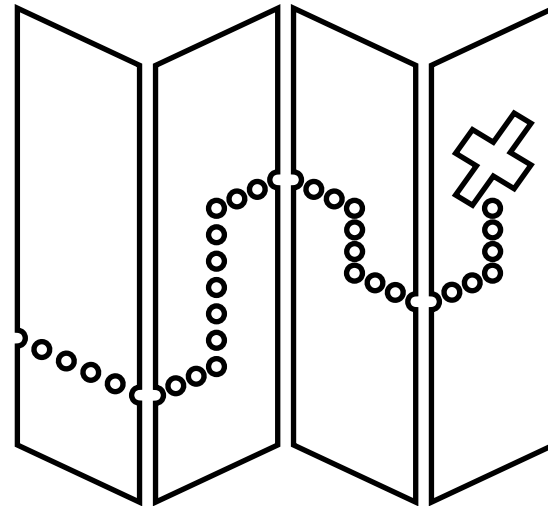
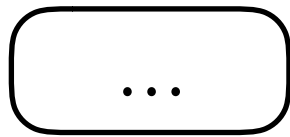
CAN NOT BE EMPTY!

WATCHES A REFERENCE

Observations

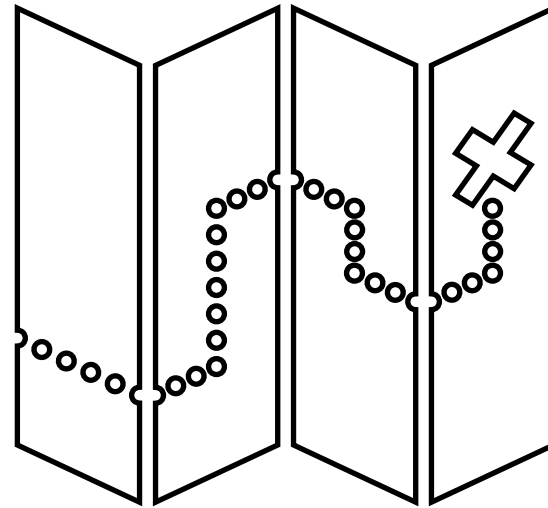
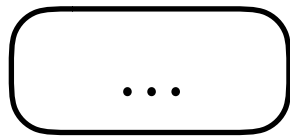


Observations



$$\mathcal{V}(\Box \text{True}) = \text{True}$$

Observations

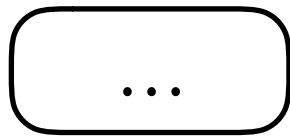


$$\mathcal{V}(\Box(51^{\circ}49'N, 5^{\circ}52'E)) \\ = (51^{\circ}49'N, 5^{\circ}52'E)$$

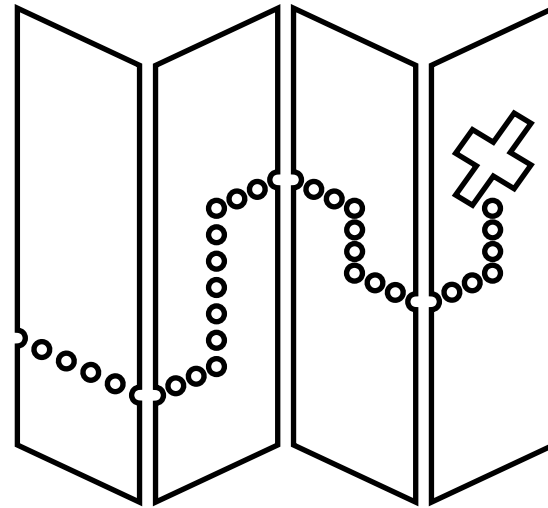


$$\mathcal{V}(\Box\text{True}) = \text{True}$$

Observations



$$\mathcal{V}(\Box \text{INT}) = \perp$$

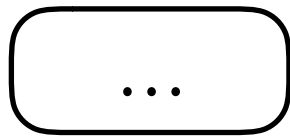


$$\begin{aligned} \mathcal{V}(\Box(51^\circ 49' \text{N}, 5^\circ 52' \text{E})) \\ = (51^\circ 49' \text{N}, 5^\circ 52' \text{E}) \end{aligned}$$

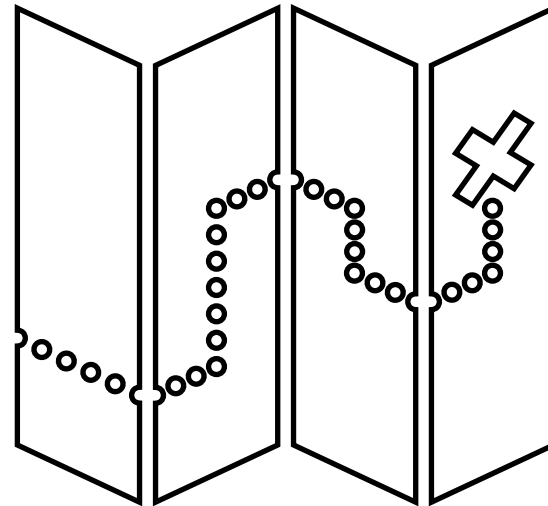


$$\mathcal{V}(\Box \text{True}) = \text{True}$$

Observations



$$\mathcal{V}(\boxtimes \text{INT}) = \perp$$



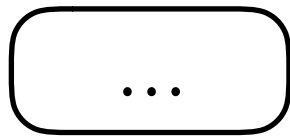
$$\begin{aligned} \mathcal{V}(\Box(51^{\circ}49'N, 5^{\circ}52'E)) \\ = (51^{\circ}49'N, 5^{\circ}52'E) \end{aligned}$$



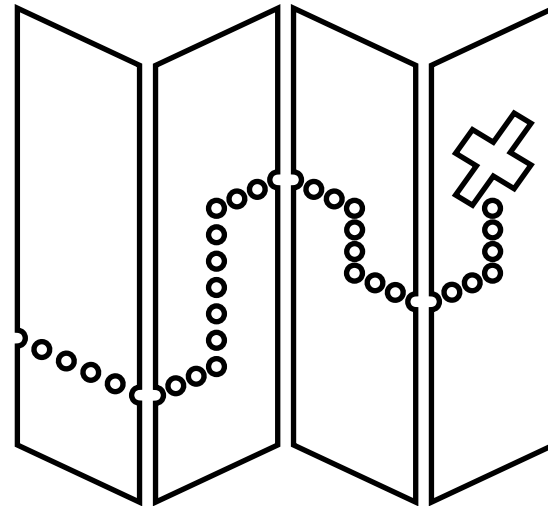
$$\mathcal{V}(\Box \text{True}) = \text{True}$$

$$\mathcal{V} : \text{TASK } \tau \longrightarrow \text{MAYBE } \tau$$

Observations



$$\mathcal{V}(\boxtimes \text{INT}) = \perp$$



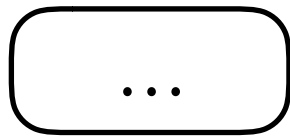
$$\begin{aligned} \mathcal{V}(\Box(51^{\circ}49'N, 5^{\circ}52'E)) \\ = (51^{\circ}49'N, 5^{\circ}52'E) \end{aligned}$$



$$\mathcal{V}(\Box \text{True}) = \text{True}$$

value $\mathcal{V} : \text{TASK } \tau \rightarrow \text{MAYBE } \tau$

Observations

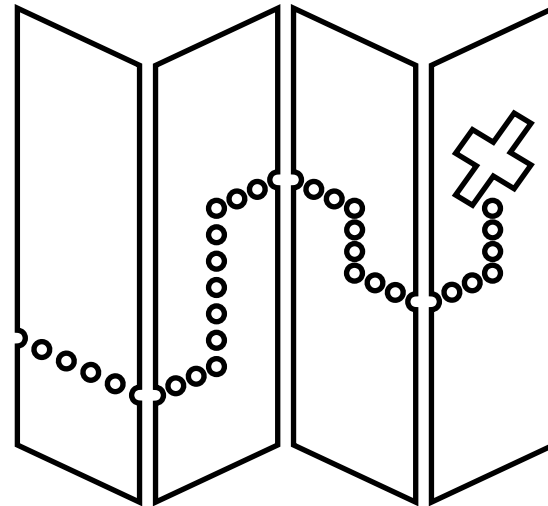


$$\mathcal{V}(\Box \text{INT}) = \perp$$

SOME TASKS DO NOT HAVE A VALUE

value

$$\mathcal{V} : \text{TASK } \tau \rightarrow \text{MAYBE } \tau$$

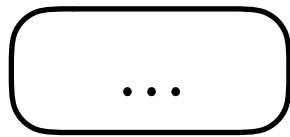


$$\begin{aligned} \mathcal{V}(\Box(51^\circ 49' \text{N}, 5^\circ 52' \text{E})) \\ = (51^\circ 49' \text{N}, 5^\circ 52' \text{E}) \end{aligned}$$



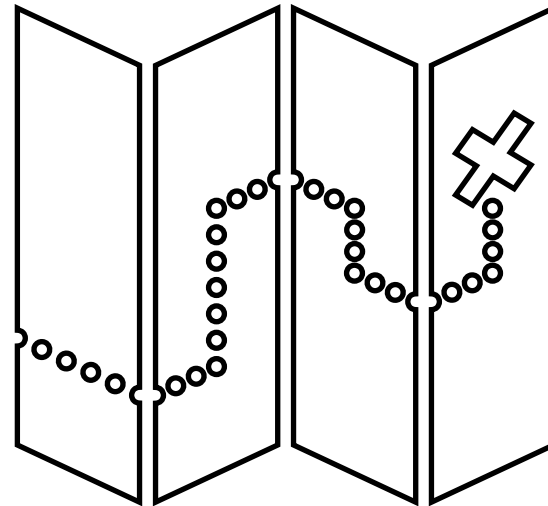
$$\mathcal{V}(\Box \text{True}) = \text{True}$$

Observations



$$\mathcal{V}(\boxtimes \text{INT}) = \perp$$

SOME TASKS DO NOT HAVE A VALUE



$$\begin{aligned} \mathcal{V}(\Box(51^{\circ}49'N, 5^{\circ}52'E)) \\ = (51^{\circ}49'N, 5^{\circ}52'E) \end{aligned}$$

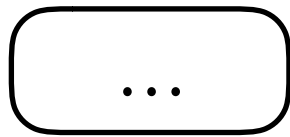


$$\mathcal{V}(\Box \text{True}) = \text{True}$$

value $\mathcal{V} : \text{TASK } \tau \rightarrow \text{MAYBE } \tau$

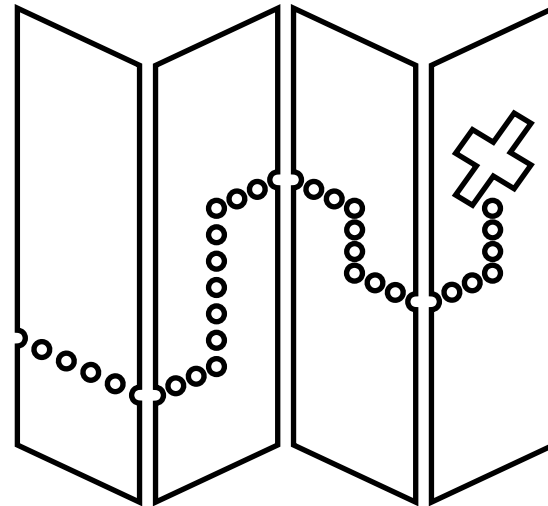
user interface $\mathcal{U} : \text{TASK } \tau \rightarrow \text{HTML}$ (or $\mathcal{U} : \text{TASK } \tau \rightarrow \text{STRING}$ or $\mathcal{U} : \text{TASK } \tau \rightarrow \dots$)

Observations



$$\mathcal{V}(\boxtimes \text{INT}) = \perp$$

SOME TASKS DO NOT HAVE A VALUE



$$\begin{aligned} \mathcal{V}(\Box(51^{\circ}49'N, 5^{\circ}52'E)) \\ = (51^{\circ}49'N, 5^{\circ}52'E) \end{aligned}$$



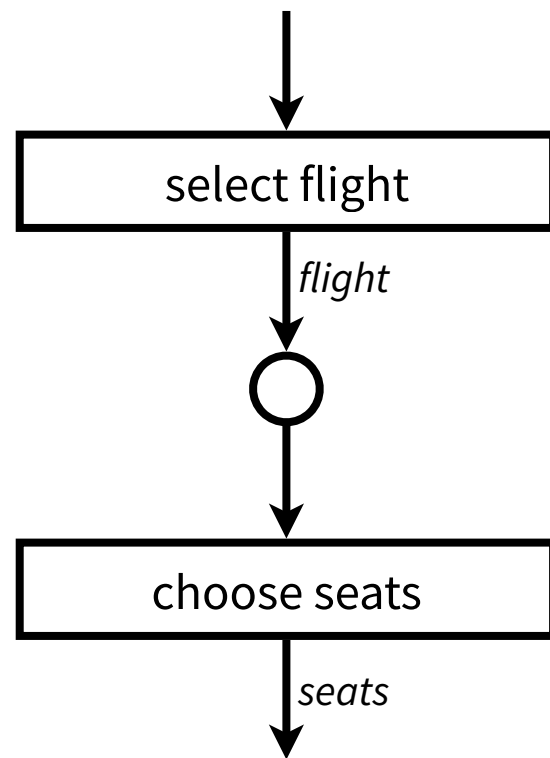
$$\mathcal{V}(\Box \text{True}) = \text{True}$$

value $\mathcal{V} : \text{TASK } \tau \rightarrow \text{MAYBE } \tau$

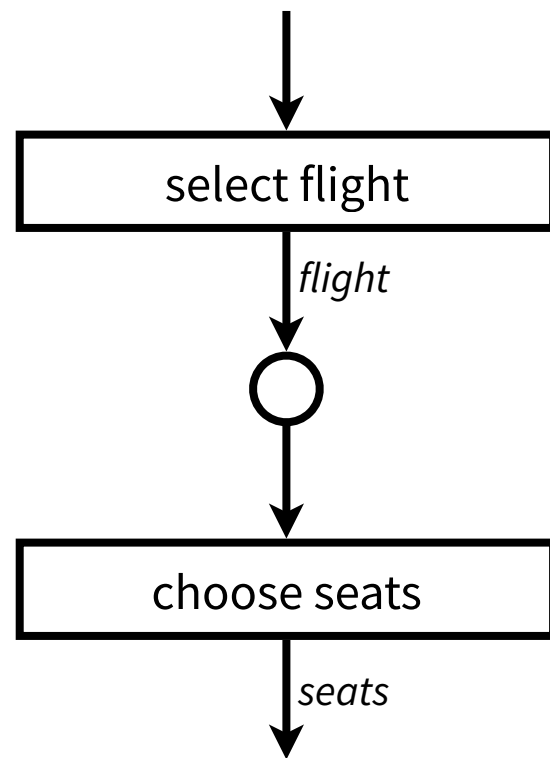
user interface $\mathcal{U} : \text{TASK } \tau \rightarrow \text{HTML}$ (or $\mathcal{U} : \text{TASK } \tau \rightarrow \text{STRING}$ or $\mathcal{U} : \text{TASK } \tau \rightarrow \dots$)

possible inputs $\mathcal{J} : \text{TASK } \tau \rightarrow \text{LIST INPUT}$

Steps

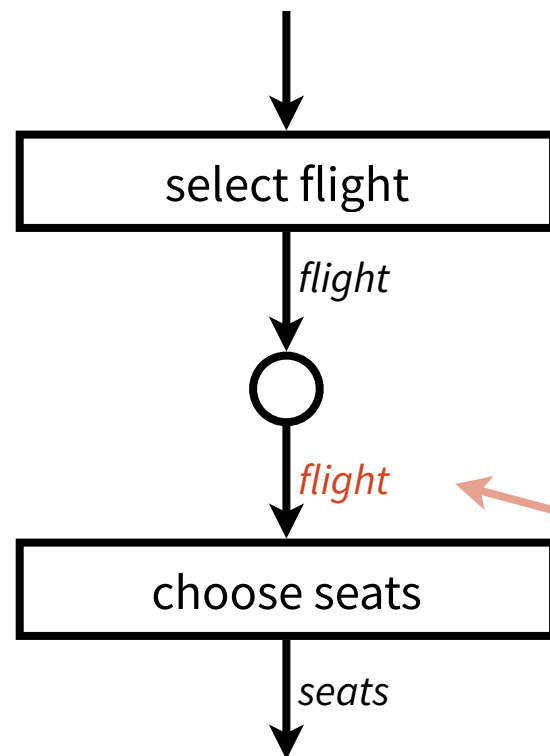


Steps



select_flight ► choose_seats

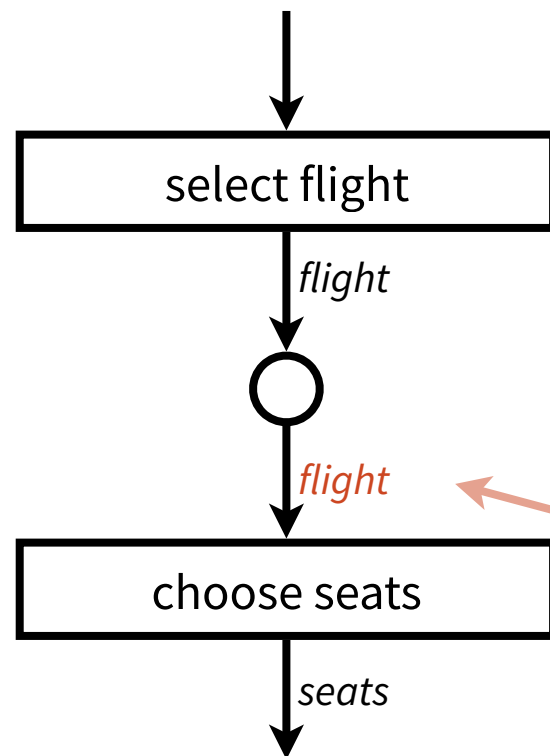
Steps



select_flight ► choose_seats

USE VALUE IN NEXT TASK?

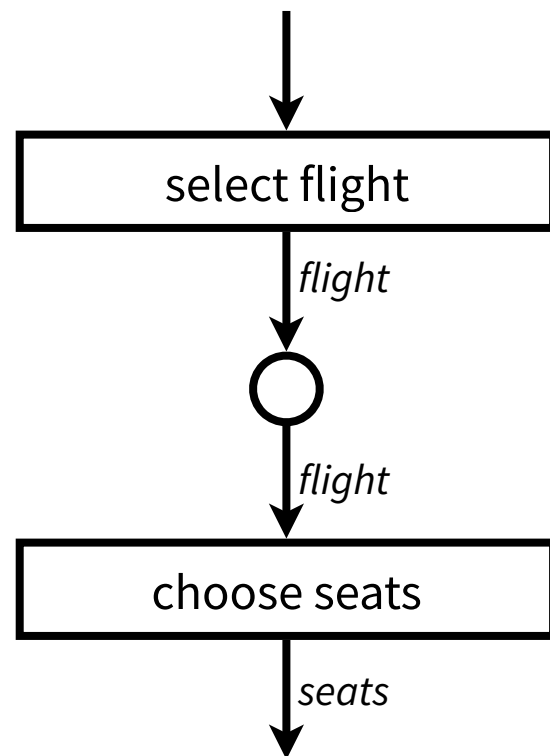
Steps



select_flight ► $\lambda flight.$ choose_seats *flight*

USE VALUE IN NEXT TASK?

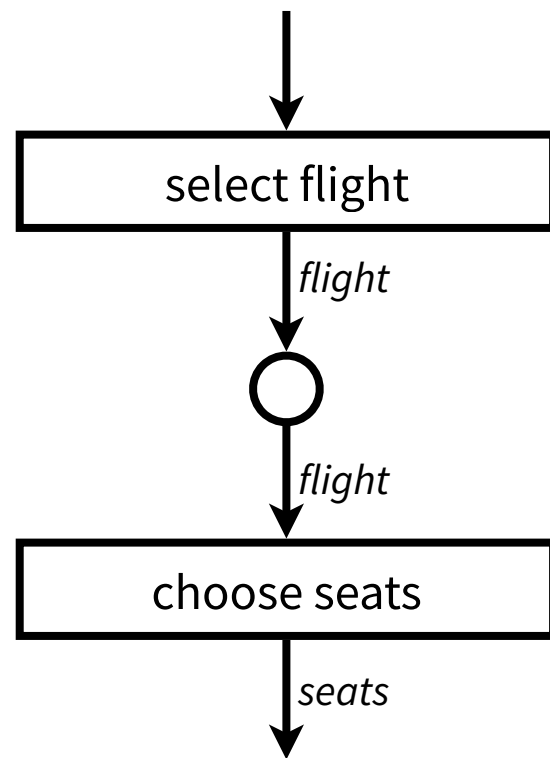
Steps



`select_flight` \triangleright $\lambda flight. \text{choose_seats } flight$

WHEN TO PROCEED TO THE NEXT TASK?

Steps

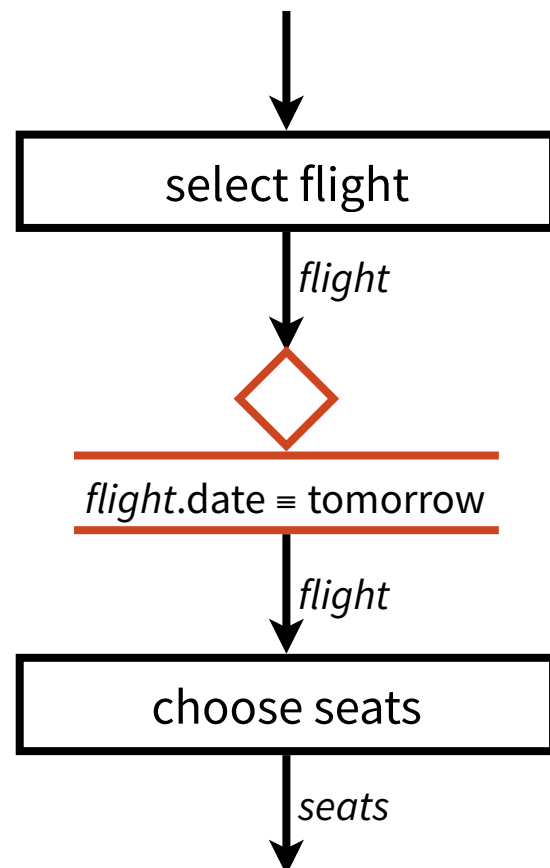


$\text{select_flight} \triangleright \lambda \text{flight}. \text{choose_seats } \text{flight}$

WHEN TO PROCEED TO THE NEXT TASK?

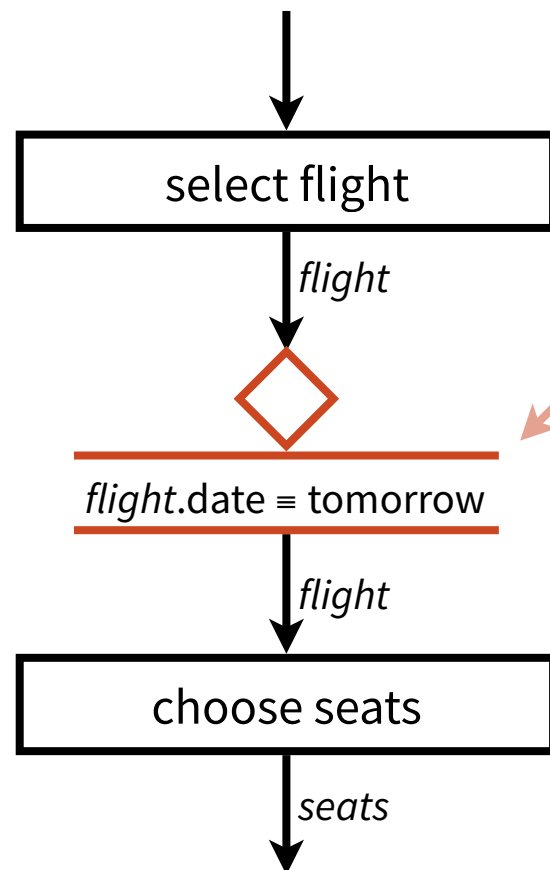
$$\Rightarrow \mathcal{V}(\text{select_flight}) = v$$

Guarded steps



`select_flight` ▶ $\lambda flight. \text{choose_seats } flight$

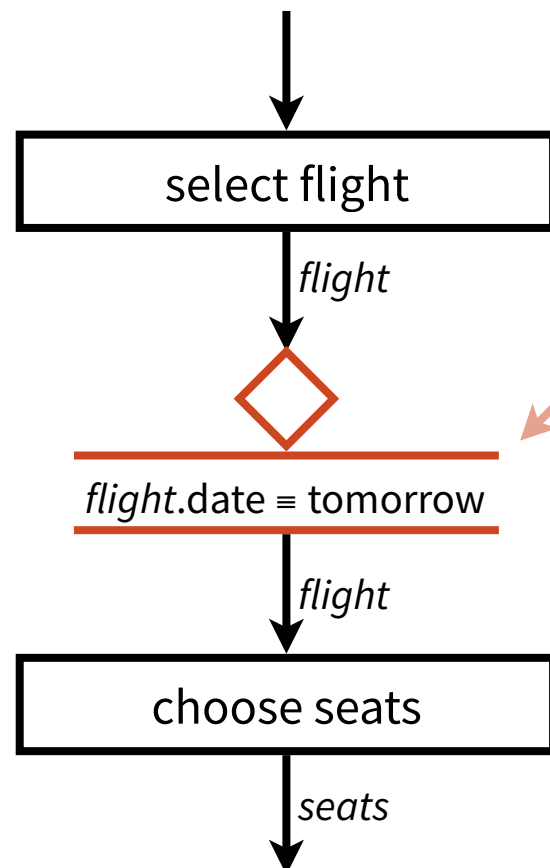
Guarded steps



ONLY WHEN GUARD IS TRUE?

$select_flight \triangleright \lambda flight. choose_seats\ flight$

Guarded steps



ONLY WHEN GUARD IS TRUE?

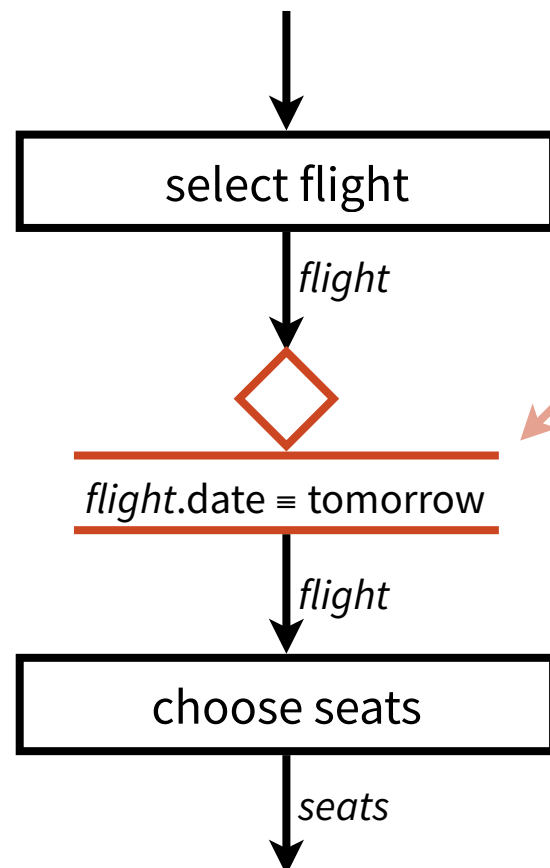
`select_flight` ▶ $\lambda flight.$

if `flight.date` \equiv `tomorrow`

then `choose_seats` `flight`

else ↯

Guarded steps



ONLY WHEN GUARD IS TRUE?

`select_flight` $\triangleright \lambda flight.$

if `flight.date == tomorrow`

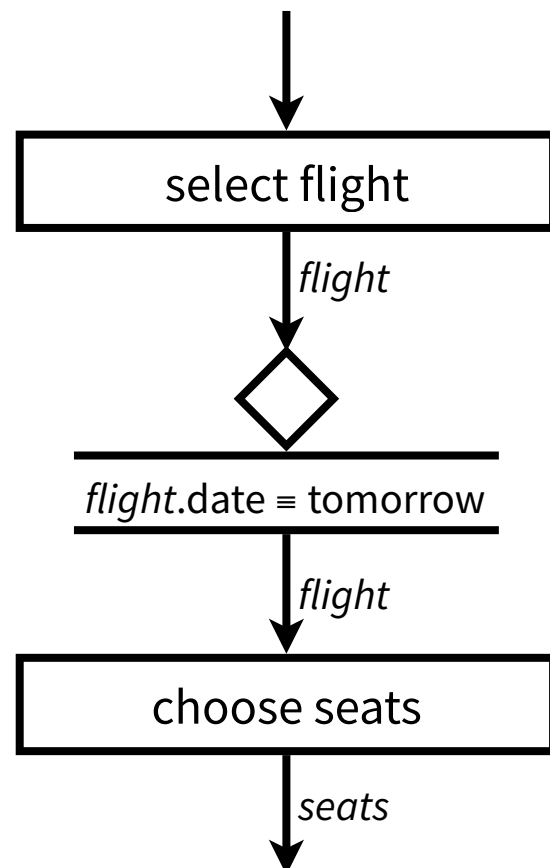
then `choose_seats flight`

else \Downarrow

FAILURE:

A TASK THAT NEVER ENDS
AND CAN NOT HANDLE USER INPUT

Guarded steps

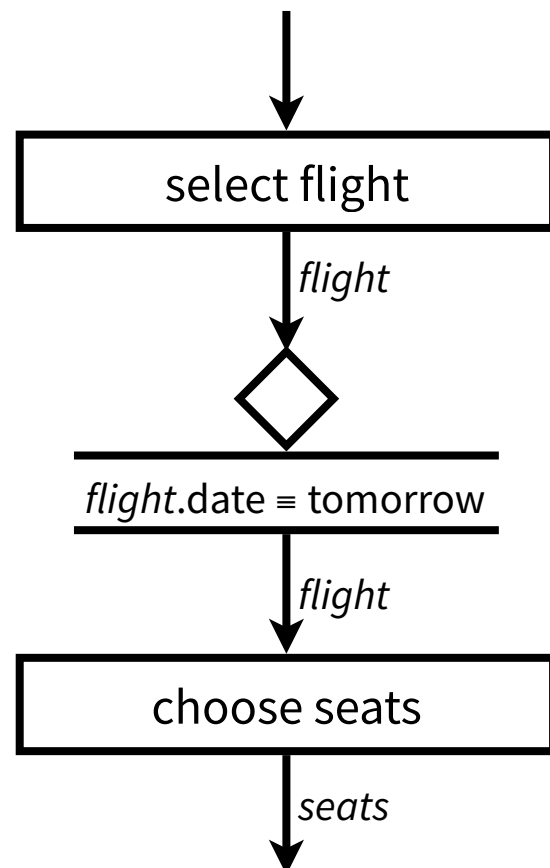


select_flight $\triangleright \lambda flight.$
if flight.date \equiv tomorrow
then choose_seats flight
else \downarrow

WHEN TO PROCEED TO THE NEXT TASK?

$$\Rightarrow \mathcal{V}(\text{select_flight}) = v$$

Guarded steps



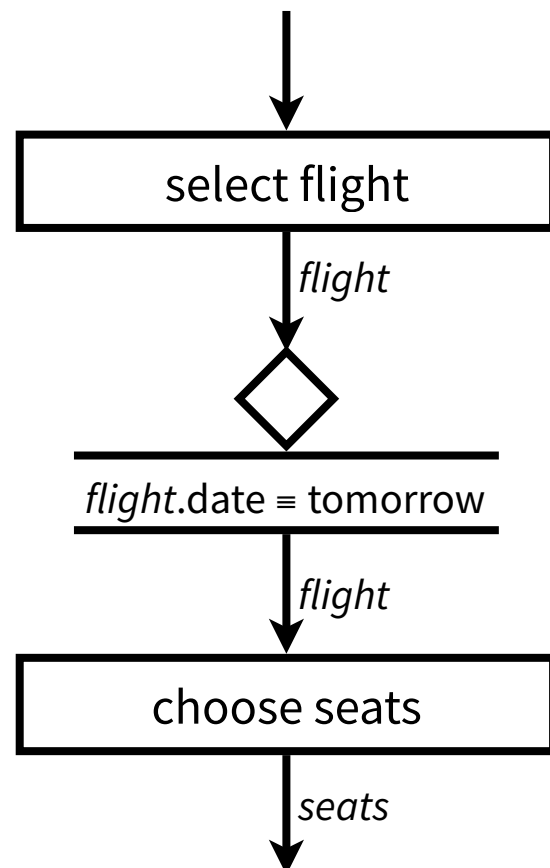
$\text{select_flight} \triangleright \lambda \text{flight}.$
if $\text{flight.date} \equiv \text{tomorrow}$
then $\text{choose_seats } \text{flight}$
else \downarrow

WHEN TO PROCEED TO THE NEXT TASK?

$\Rightarrow \mathcal{V}(\text{select_flight}) = v$

and $\dots (v) \Downarrow t$ **where** $t \neq \downarrow$

Guarded steps



select_flight $\triangleright \lambda flight.$
 if flight.date \equiv tomorrow
 then choose_seats flight
 else \downarrow

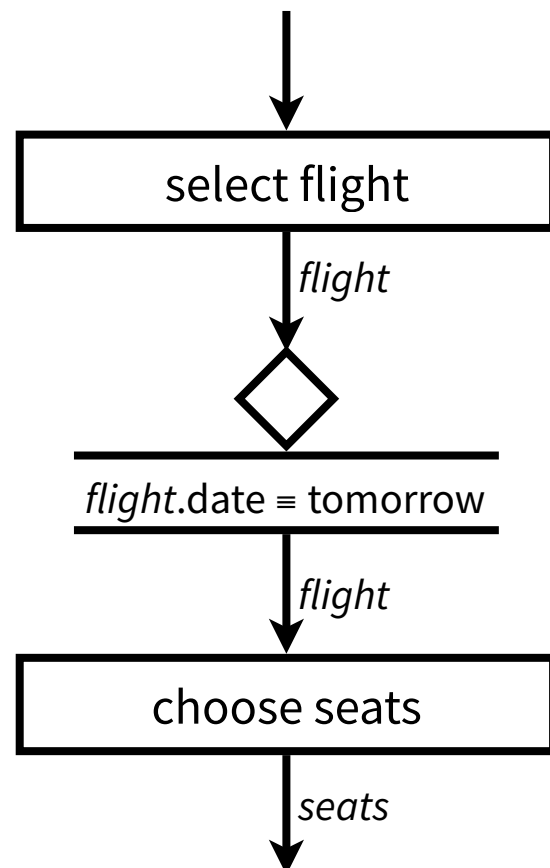
WHEN TO PROCEED TO THE NEXT TASK?

$\Rightarrow \mathcal{V}(\text{select_flight}) = v$

and $\dots (v) \Downarrow t$ where $t \neq \downarrow$

USING HOST LANGUAGE SEMANTICS!

Guarded steps



$\text{select_flight} \triangleright \lambda \text{flight}.$
if $\text{flight.date} \equiv \text{tomorrow}$
then $\text{choose_seats } \text{flight}$
else \downarrow

WHEN TO PROCEED TO THE NEXT TASK?

$\Rightarrow \mathcal{V}(\text{select_flight}) = v$
and $\dots (v) \Downarrow t$ **where** $t \neq \downarrow$



USING HOST LANGUAGE SEMANTICS!



Nitty gritty

Grammar

Take a λ -calculus...

$e ::=$		Expressions		
$\lambda x : \tau. e$	$e_1 e_2$	– abstraction, application		
x	c	$e_1 \star e_2$	– variable, constant, operation	
if e_1 then e_2 else e_3	$\langle \rangle$	– branch, unit		
$\langle e_1, e_2 \rangle$	fst e	snd e	– pair, projections	
ref e	$!e$	$e_1 := e_2$	l	– references, location
p				– pretask
$c ::=$		Constants		
B	I	S	– boolean, integer, string	

Grammar

Take a λ -calculus...

$e ::=$	Expressions
$\lambda x : \tau. e$ $e_1 e_2$	– abstraction, application
x c $e_1 \star e_2$	– variable, constant, operation
if e_1 then e_2 else e_3 $\langle \rangle$	– branch, unit
$\langle e_1, e_2 \rangle$ fst e snd e	– pair, projections
ref e $!e$ $e_1 := e_2$ l	– references, location
p	– pretask
$c ::=$	Constants
B I S	– boolean, integer, string

...embed a workflow language

$t ::=$	Tasks
$\square v$ $\boxtimes \tau$ $\blacksquare l$	– editors
$t_1 \blacktriangleright e_2$ $t_1 \triangleright e_2$	– steps
$\not\downarrow$ $t_1 \bowtie t_2$	– fail, combination
$t_1 \blacklozenge t_2$ $e_1 \lozenge e_2$	– choices

Grammar

Take a λ -calculus...

$e ::=$	Expressions
$\lambda x : \tau. e$ $e_1 e_2$	– abstraction, application
x c $e_1 \star e_2$	– variable, constant, operation
if e_1 then e_2 else e_3 $\langle \rangle$	– branch, unit
$\langle e_1, e_2 \rangle$ fst e snd e	– pair, projections
ref e $!e$ $e_1 := e_2$ l	– references, location
p	– pretask
$c ::=$	Constants
B I S	– boolean, integer, string

...embed a workflow language

$t ::=$	Tasks
$\square v$ $\boxtimes \tau$ $\blacksquare l$	– editors
$t_1 \blacktriangleright e_2$ $t_1 \triangleright e_2$	– steps
$\not\downarrow$ $t_1 \bowtie t_2$	– fail, combination
$t_1 \blacklozenge t_2$ $e_1 \lozenge e_2$	– choices

COMBINATION OF TWO TASKS

Grammar

Take a λ -calculus...

$e ::=$	Expressions
$\lambda x : \tau. e$ $e_1 e_2$	– abstraction, application
x c $e_1 \star e_2$	– variable, constant, operation
if e_1 then e_2 else e_3 $\langle \rangle$	– branch, unit
$\langle e_1, e_2 \rangle$ fst e snd e	– pair, projections
ref e $!e$ $e_1 := e_2$ l	– references, location
p	– pretask
$c ::=$	Constants
B I S	– boolean, integer, string

...embed a workflow language

$t ::=$	Tasks
$\square v$ $\boxtimes \tau$ $\blacksquare l$	– editors
$t_1 \blacktriangleright e_2$ $t_1 \triangleright e_2$	– steps
$\not\downarrow$ $t_1 \bowtie t_2$	– fail, combination
$t_1 \blacklozenge t_2$ $e_1 \diamond e_2$	– choices

COMBINATION OF TWO TASKS

CHOICE BETWEEN TWO TASKS

Semantics

Two layers \Rightarrow two semantics

Semantics

Two layers \Rightarrow two semantics

$e \downarrow v$ **STANDARD BIG STEP SEMANTICS**

Semantics

Two layers \Rightarrow two semantics

$e \downarrow v$ STANDARD BIG STEP SEMANTICS

$p \rightsquigarrow t$ SPECIAL TASK SEMANTICS

Semantics

Two layers \Rightarrow two semantics



Semantics

Two layers \Rightarrow two semantics



$$\frac{\text{S-THENSTAY} \quad t_1, s \rightsquigarrow t'_1, s'}{t_1 \blacktriangleright e_2, s \rightsquigarrow t'_1 \blacktriangleright e_2, s'} \quad \mathcal{V}(t'_1, s') = \perp$$

$$\frac{\text{S-THENFAIL} \quad t_1, s \rightsquigarrow t'_1, s' \quad e_2 \ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t'_1 \blacktriangleright e_2, s'} \quad \mathcal{V}(t'_1, s') = v_1 \wedge \mathcal{F}(t_2, s'')$$

$$\frac{\text{S-THENCONT} \quad t_1, s \rightsquigarrow t'_1, s' \quad e_2 \ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_2, s'''} \quad \mathcal{V}(t'_1, s') = v_1 \wedge \neg \mathcal{F}(t_2, s'')$$

Semantics

Two layers \Rightarrow ^{three} ~~two~~ semantics

$e \downarrow v$ **STANDARD BIG STEP SEMANTICS**

$p \rightsquigarrow t$ **SPECIAL TASK SEMANTICS**

But interaction... \Rightarrow additional layer

$$\frac{\text{S-THENSTAY} \quad t_1, s \rightsquigarrow t'_1, s'}{t_1 \blacktriangleright e_2, s \rightsquigarrow t'_1 \blacktriangleright e_2, s'} \quad \mathcal{V}(t'_1, s') = \perp$$

$$\frac{\text{S-THENFAIL} \quad t_1, s \rightsquigarrow t'_1, s' \quad e_2 \ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t'_1 \blacktriangleright e_2, s'} \quad \mathcal{V}(t'_1, s') = v_1 \wedge \mathcal{F}(t_2, s'')$$

$$\frac{\text{S-THENCONT} \quad t_1, s \rightsquigarrow t'_1, s' \quad e_2 \ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_2, s'''} \quad \mathcal{V}(t'_1, s') = v_1 \wedge \neg \mathcal{F}(t_2, s'')$$

Semantics

Two layers \Rightarrow ^{three} ~~two~~ semantics

$e \downarrow v$ **STANDARD BIG STEP SEMANTICS**

$p \rightsquigarrow t$ **SPECIAL TASK SEMANTICS**

But interaction... \Rightarrow additional layer

$t \xrightarrow{i} t'$ **HANDLING OF USER INPUT**

$$\frac{\text{S-THENSTAY} \quad t_1, s \rightsquigarrow t'_1, s'}{t_1 \blacktriangleright e_2, s \rightsquigarrow t'_1 \blacktriangleright e_2, s'} \quad \mathcal{V}(t'_1, s') = \perp$$

$$\frac{\text{S-THENFAIL} \quad t_1, s \rightsquigarrow t'_1, s' \quad e_2 \ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t'_1 \blacktriangleright e_2, s'} \quad \mathcal{V}(t'_1, s') = v_1 \wedge \mathcal{F}(t_2, s'')$$

$$\frac{\text{S-THENCONT} \quad t_1, s \rightsquigarrow t'_1, s' \quad e_2 \ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_2, s'''} \quad \mathcal{V}(t'_1, s') = v_1 \wedge \neg \mathcal{F}(t_2, s'')$$

Semantics

Two layers \Rightarrow ^{three} ~~two~~ semantics

$e \downarrow v$ **STANDARD BIG STEP SEMANTICS**

$p \rightsquigarrow t$ **SPECIAL TASK SEMANTICS**

But interaction... \Rightarrow additional layer

$t \xrightarrow{i} t'$ **HANDLING OF USER INPUT**

$$\frac{\text{S-THENSTAY} \quad t_1, s \rightsquigarrow t'_1, s'}{t_1 \blacktriangleright e_2, s \rightsquigarrow t'_1 \blacktriangleright e_2, s'} \quad \mathcal{V}(t'_1, s') = \perp$$

$$\frac{\text{S-THENFAIL} \quad t_1, s \rightsquigarrow t'_1, s' \quad e_2 \ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t'_1 \blacktriangleright e_2, s'} \quad \mathcal{V}(t'_1, s') = v_1 \wedge \mathcal{F}(t_2, s'')$$

$$\frac{\text{S-THENCONT} \quad t_1, s \rightsquigarrow t'_1, s' \quad e_2 \ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_2, s'''} \quad \mathcal{V}(t'_1, s') = v_1 \wedge \neg \mathcal{F}(t_2, s'')$$

□5

$\xrightarrow{\text{Empty}} \boxtimes \text{INT}$

■(51°49'N, 5°52'E)

$\xrightarrow{(51°49'N, 5°52'W)} \blacksquare(51°49'N, 5°52'W)$



= the essence of task oriented programming



= the essence of task oriented programming

Language + Formal semantics

$t ::=$

| $\square v$ | $\boxtimes \tau$ | $\blacksquare l$
| $t_1 \blacktriangleright e_2$ | $t_1 \triangleright e_2$
| $\not\downarrow$ | $t_1 \bowtie t_2$
| $t_1 \blacklozenge t_2$ | $e_1 \lozenge e_2$

Tasks

- editors
- steps
- fail, combination
- choices





= the essence of task oriented programming

Language + Formal semantics

$t ::=$

| $\square v$ | $\boxtimes \tau$ | $\blacksquare l$
| $t_1 \blacktriangleright e_2$ | $t_1 \triangleright e_2$
| $\not\downarrow$ | $t_1 \bowtie t_2$
| $t_1 \blacklozenge t_2$ | $e_1 \lozenge e_2$

Tasks

- editors
- steps
- fail, combination
- choices

Proved progress & preservation





= the essence of task oriented programming

Language + Formal semantics

$t ::=$

| $\square v$ | $\boxtimes \tau$ | $\blacksquare l$
| $t_1 \blacktriangleright e_2$ | $t_1 \triangleright e_2$
| $\not\downarrow$ | $t_1 \bowtie t_2$
| $t_1 \blacklozenge t_2$ | $e_1 \lozenge e_2$

Tasks

- editors
- steps
- fail, combination
- choices

Proved progress & preservation

Implemented in Idris





= the essence of task oriented programming

Language + Formal semantics

$t ::=$

| $\square v$ | $\boxtimes \tau$ | $\blacksquare l$
| $t_1 \blacktriangleright e_2$ | $t_1 \triangleright e_2$
| $\not\downarrow$ | $t_1 \bowtie t_2$
| $t_1 \blacklozenge t_2$ | $e_1 \lozenge e_2$

Tasks

- editors
- steps
- fail, combination
- choices

Proved progress & preservation

Implemented in Idris

Still to do...

- Task equality
- Pre- and postconditions
- Symbolic execution



= the essence of task oriented programming

Language + Formal semantics

$t ::=$

| $\square v$ | $\boxtimes \tau$ | $\blacksquare l$
| $t_1 \blacktriangleright e_2$ | $t_1 \triangleright e_2$
| $\not\downarrow$ | $t_1 \bowtie t_2$
| $t_1 \blacklozenge t_2$ | $e_1 \lozenge e_2$

Tasks

- editors
- steps
- fail, combination
- choices

Proved progress & preservation

Implemented in Idris

Still to do...

- Task equality
- Pre- and postconditions
- Symbolic execution

IDEAS APPRECIATED!





Thank you

Summary



Language & formal semantics



Proved progress & preservation



Implemented in Idris



Essence of task oriented programming

Grammar

TAKE A λ -CALCULUS...

$e ::=$
 $\lambda x : \tau. e \mid e_1 e_2$
 $x \mid c \mid e_1 \star e_2$
 $\text{if } e_1 \text{ then } e_2 \text{ else } e_3 \mid \langle \rangle$
 $\langle e_1, e_2 \rangle \mid \text{fst } e \mid \text{snd } e$
 $\text{ref } e \mid !e \mid e_1 := e_2 \mid l$
 p
 $c ::=$
 $B \mid I \mid S$

Expressions

- abstraction, application
- variable, constant, operation
- branch, unit
- pair, projections
- references, location
- pretask

Constants

- boolean, integer, string

...EMBED A WORKFLOW LANGUAGE

$t ::=$

$\square v \mid \boxtimes \tau \mid \blacksquare l$
 $t_1 \blacktriangleright e_2 \mid t_1 \triangleright e_2$
 $\downarrow \mid t_1 \bowtie t_2$
 $t_1 \blacklozenge t_2 \mid e_1 \diamond e_2$

Tasks

- editors
- steps
- fail, combination
- choices

RACE BETWEEN TWO TASKS

COMBINATION OF TWO TASKS



Summary

- Language for modular interactive workflows
- Essence of task oriented programming
- Formal semantics
- Proved progress & preservation
- Implemented in Idris

$t ::=$

| $\square v$ | $\boxtimes \tau$ | $\blacksquare l$
| $t_1 \blacktriangleright e_2$ | $t_1 \triangleright e_2$
| $\not\downarrow$ | $t_1 \bowtie t_2$
| $t_1 \blacklozenge t_2$ | $e_1 \lozenge e_2$

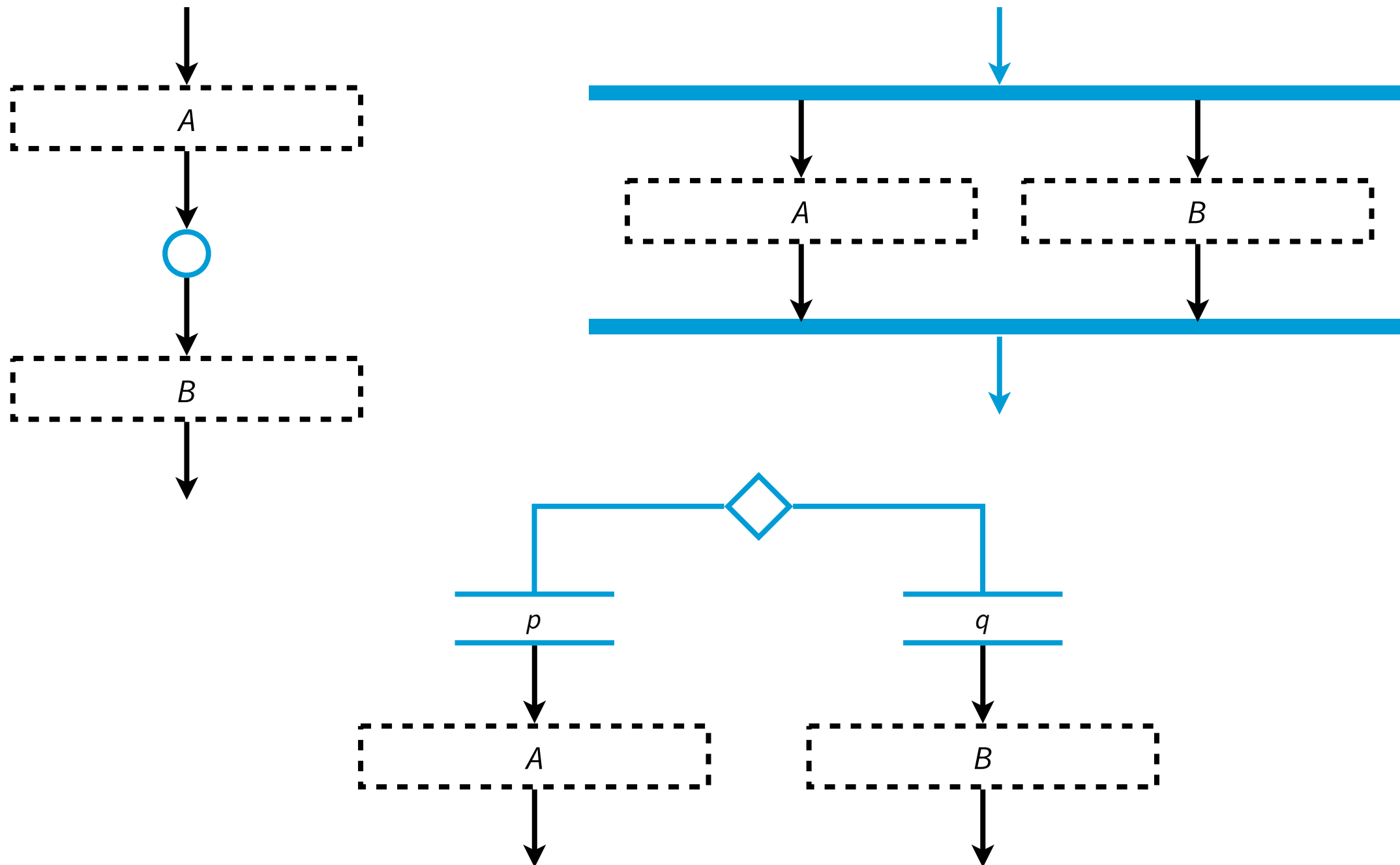
Tasks

- editors
- steps
- fail, combination
- choices

LANGUAGE
+
FORMAL SEMANTICS



Tasks



l , \boxtimes \square \boxminus \blacksquare \square \blacksquare \square \blacksquare \square \blacksquare \square

l , \boxtimes \square \boxminus \blacksquare \square \blacksquare \square \blacksquare \square \blacksquare \square

\blacksquare l , \boxtimes \square \boxminus \blacksquare \square \blacksquare \square \blacksquare \square \blacksquare \square

