TopHat: a stylish journey through modular interactive workflows

Tim Steenvoorden
Markus Klinik
Nico Naus

Dutch FP Day, January 11, 2019

Institute for Computing and Information Sciences
Radboud University

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)

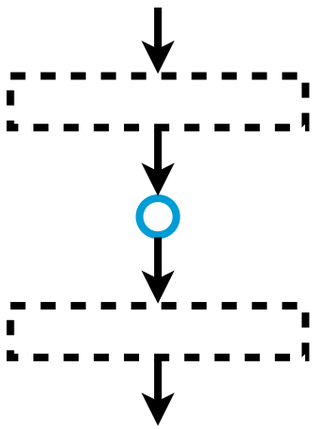
Foundation for formal reasoning
and comparison to other frameworks

Institute for Computing
and Information Sciences
Radboud University

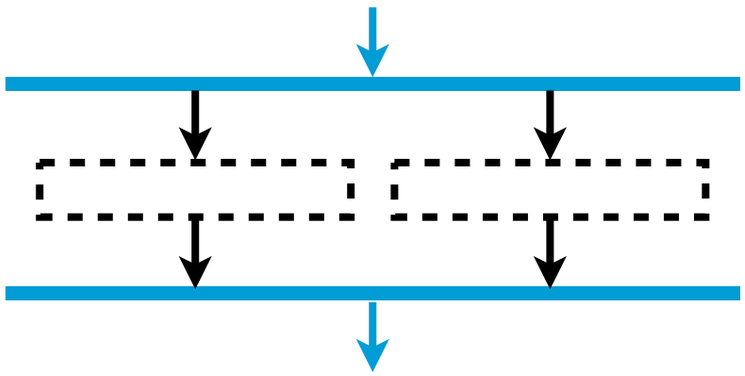
Concepts

Institute for Computing
and Information Sciences
Radboud University

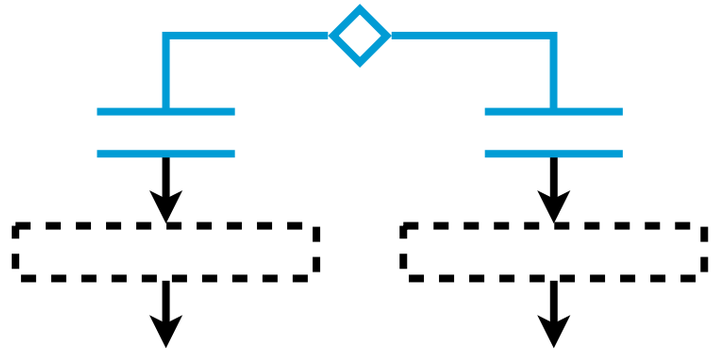
Collaboration



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*."

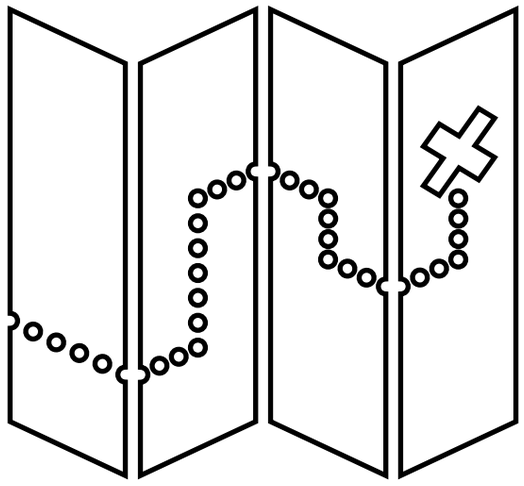
Building blocks

Institute for Computing
and Information Sciences
Radboud University

Editors

□5

Empty → ⊠INT

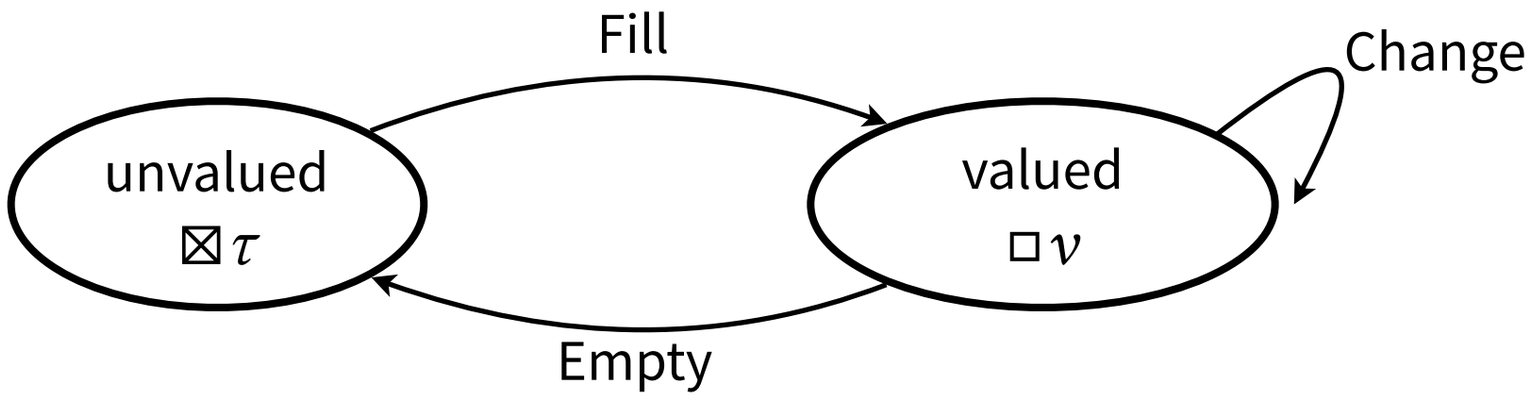


□(51°49'N, 5°52'E)

□True

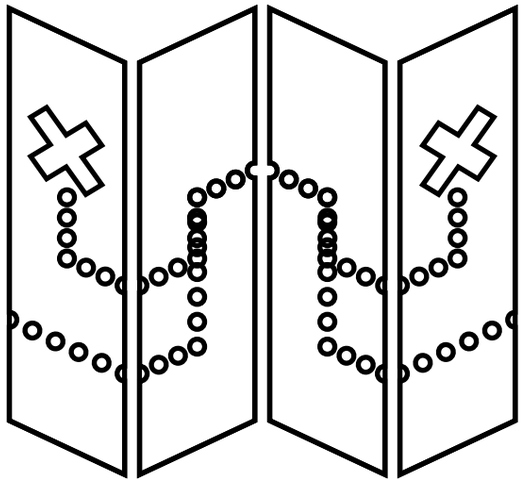
False → □False

CAPTURE THE EVER CHANGING NATURE OF WIDGETS

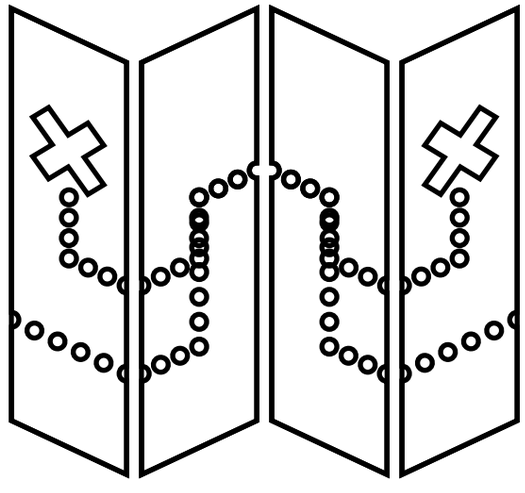


Institute for Computing and Information Sciences
Radboud University

Shared editors



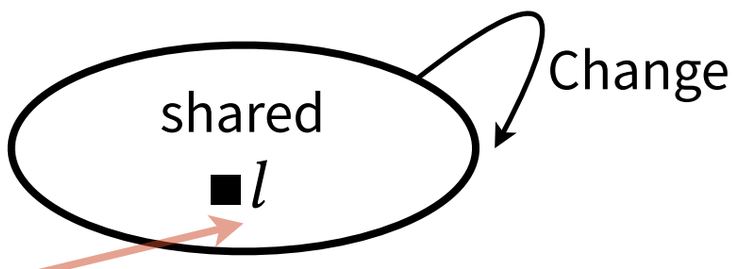
■(51°49’N, 5°52’E)



■(51°49’N, 5°52’E)

(51°49’N, 5°52’W) →

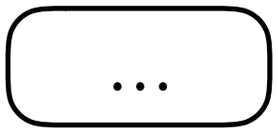
■(51°49’N, 5°52’W)



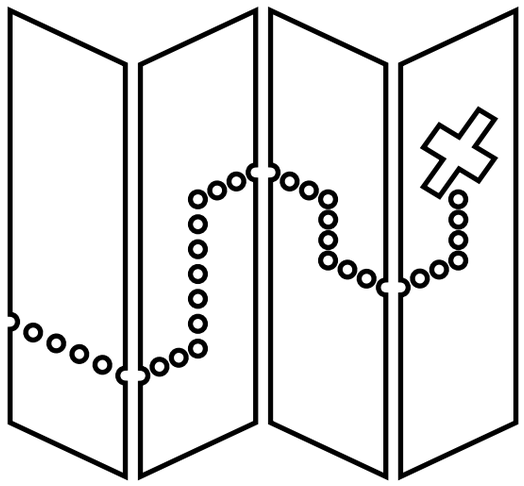
CAN NOT BE EMPTY!

WATCHES A *REFERENCE*

Observations



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



$\mathcal{V}(\square(51°49'\text{N}, 5°52'\text{E}))$
$= (51°49'\text{N}, 5°52'\text{E})$

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

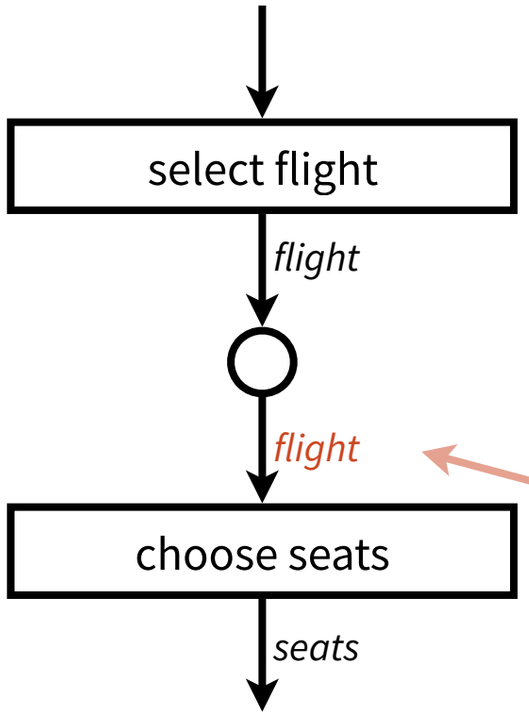
SOME TASKS DO NOT HAVE A VALUE

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

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

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

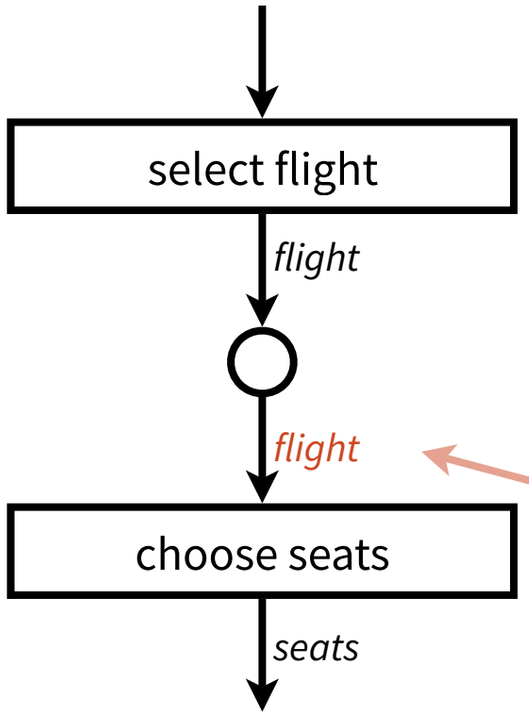
Steps



select_flight ▸ choose_seats

USE VALUE IN NEXT TASK?

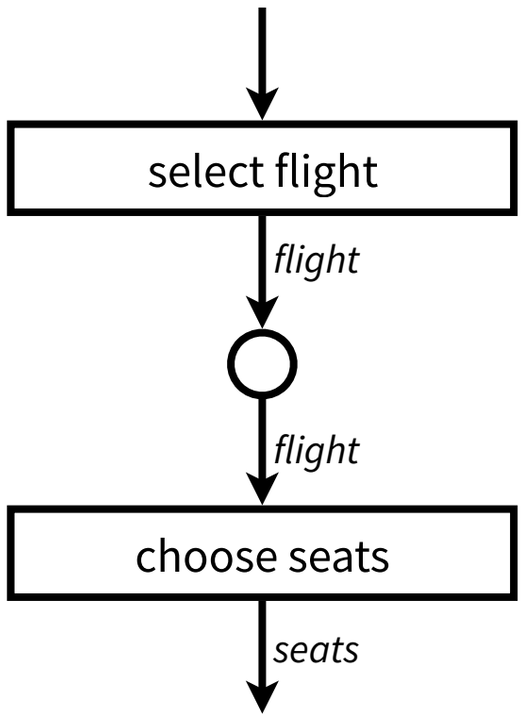
Steps



$$\text{select\_flight} \blacktriangleright \lambda \mathit{flight}.\ \text{choose\_seats}\ \mathit{flight}$$

USE VALUE IN NEXT TASK?

Steps

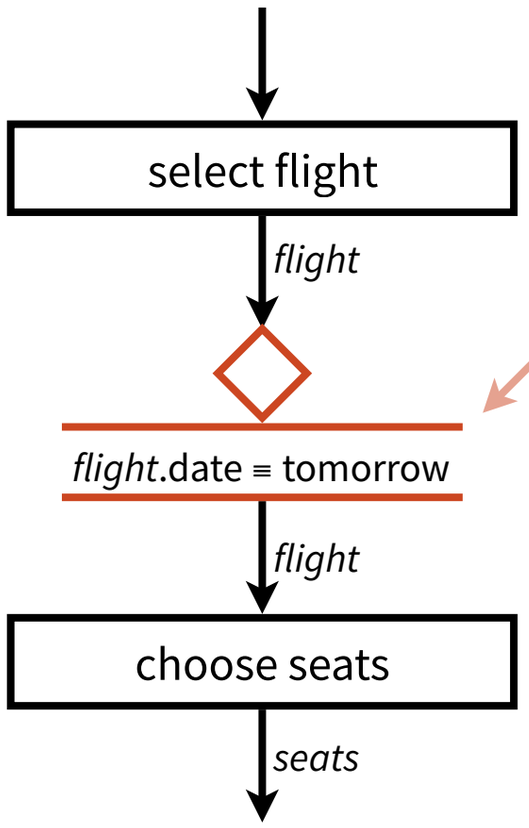


$$\text{select\_flight} \blacktriangleright \lambda \mathit{flight}.\ \text{choose\_seats}\ \mathit{flight}$$

WHEN TO PROCEED TO THE NEXT TASK?

$$\Longrightarrow \mathcal{V}(\text{select\_flight}) = v$$

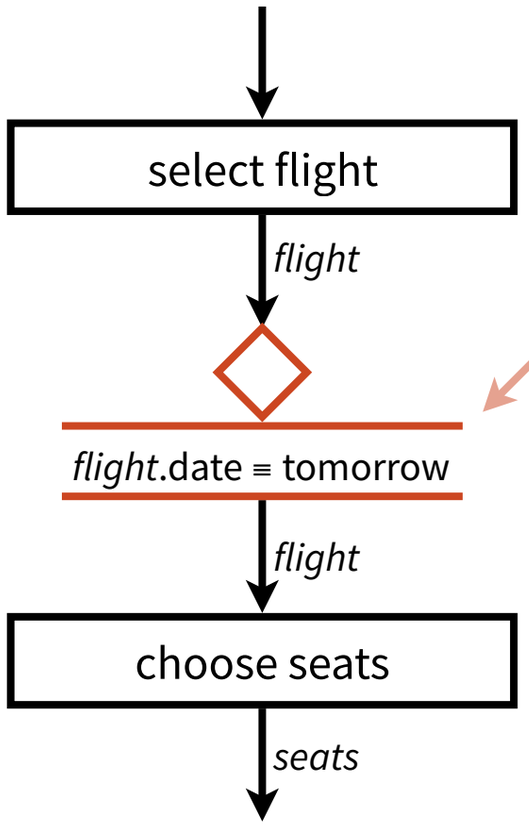
Guarded steps



ONLY WHEN GUARD IS TRUE?

select_flight ▸ λ*flight*. choose_seats *flight*

Guarded steps

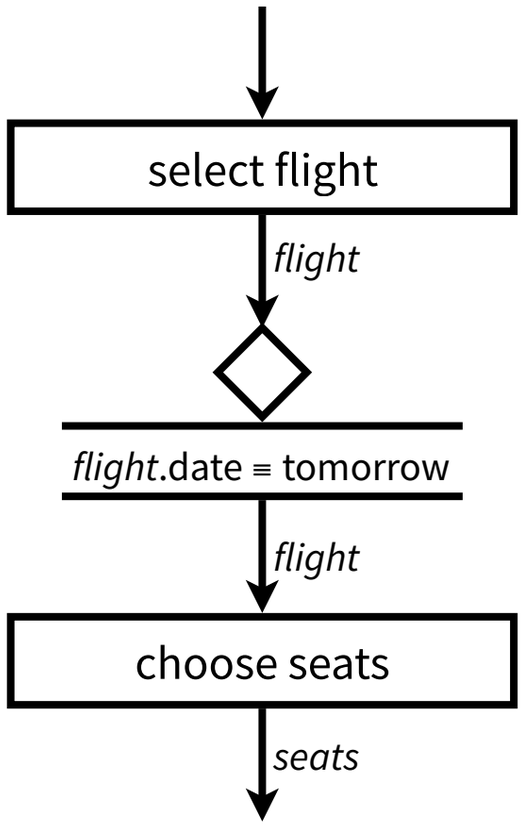


ONLY WHEN GUARD IS TRUE?

select_flight ▸ λ*flight*.
if flight.date ≡ tomorrow
then choose_seats *flight*
else ↯

FAILURE:
A TASK THAT NEVER ENDS
AND CAN NOT HANDLE USER INPUT

Guarded steps



select_flight ▸ λ*flight*.
if flight.date ≡ tomorrow
then choose_seats *flight*
else ↯

WHEN TO PROCEED TO THE NEXT TASK?

$\Longrightarrow \mathcal{V}(\text{select\_flight}) = v$
and ... $(v) \Downarrow t$ **where** $t \neq$ ↯

USING HOST LANGUAGE SEMANTICS!

Institute for Computing and Information Sciences
Radboud University

Calculus

Institute for Computing
and Information Sciences
Radboud University

Grammar

Take a λ-calculus…

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

Grammar

Take a λ-calculus…

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

…embed a workflow language

$t ::=$		Tasks
	$\square\, v \mid \boxtimes\, \tau \mid \blacksquare\, l$	– editors
	$t_1 \blacktriangleright e_2 \mid t_1 \triangleright e_2$	– steps
	$\lightning \mid t_1 \bowtie t_2$	– fail, combination
	$t_1 \blacklozenge t_2 \mid e_1 \lozenge e_2$	– choices

COMBINATION OF TWO TASKS

CHOICE BETWEEN TWO TASKS

Institute for Computing and Information Sciences
Radboud University

Semantics

Two layers ⇒ two semantics

$e \downarrow v$ STANDARD BIG STEP SEMANTICS

MAKES USE OF

$p \rightsquigarrow t$ SPECIAL TASK SEMANTICS

$$\textsc{S-ThenStay}\quad \frac{t_1, s \rightsquigarrow t_1', s'}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_1' \blacktriangleright e_2, s'}\ \mathcal{V}(t_1', s') = \bot$$

$$\textsc{S-ThenFail}\quad \frac{t_1, s \rightsquigarrow t_1', s' \qquad e_2\ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_1' \blacktriangleright e_2, s'}\ \mathcal{V}(t_1', s') = v_1 \wedge \mathcal{F}(t_2, s'')$$

$$\textsc{S-ThenCont}\quad \frac{t_1, s \rightsquigarrow t_1', s' \qquad e_2\ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_2, s'''}\ \mathcal{V}(t_1', s') = v_1 \wedge \neg\mathcal{F}(t_2, s'')$$

Semantics

Two layers ⇒ ~~two~~ three semantics

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

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

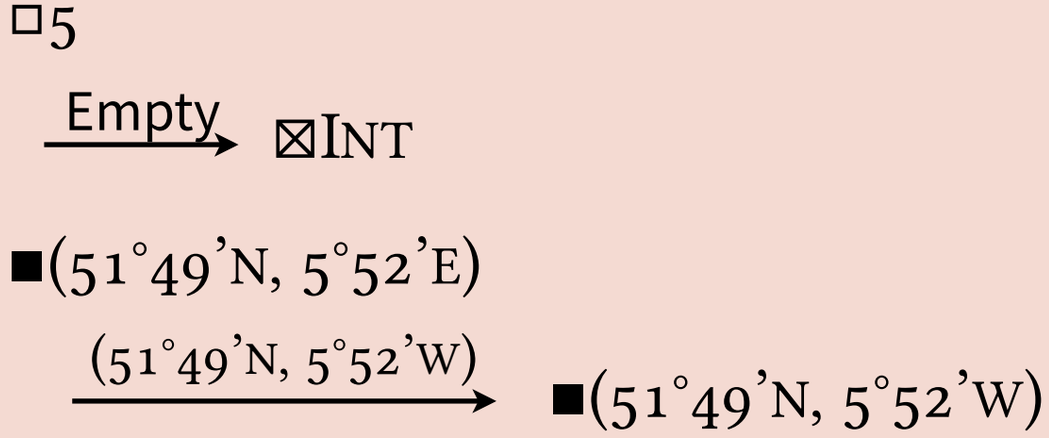
But interaction… ⇒ additional layer

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

$$\textsc{S-ThenStay}\ \frac{t_1, s \rightsquigarrow t_1', s'}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_1' \blacktriangleright e_2, s'}\ \mathcal{V}(t_1', s') = \bot$$

$$\textsc{S-ThenFail}\ \frac{t_1, s \rightsquigarrow t_1', s' \qquad e_2\ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_1' \blacktriangleright e_2, s'}\ \mathcal{V}(t_1', s') = v_1 \wedge \mathcal{F}(t_2, s'')$$

$$\textsc{S-ThenCont}\ \frac{t_1, s \rightsquigarrow t_1', s' \qquad e_2\ v_1, s' \downarrow t_2, s''}{t_1 \blacktriangleright e_2, s \rightsquigarrow t_2, s'''}\ \mathcal{V}(t_1', s') = v_1 \wedge \neg\mathcal{F}(t_2, s'')$$



Summary

Institute for Computing
and Information Sciences
Radboud University

= the essence of collaboration

Language & formal semantics

$$
\begin{array}{lll}
t ::= & & \text{Tasks} \\
\quad | & \square\, v \mid \boxtimes\, \tau \mid \blacksquare\, l & \text{– editors} \\
\quad | & t_1 \blacktriangleright e_2 \mid t_1 \triangleright e_2 & \text{– steps} \\
\quad | & \lightning \mid t_1 \bowtie t_2 & \text{– fail, combination} \\
\quad | & t_1 \blacklozenge t_2 \mid e_1 \lozenge e_2 & \text{– choices}
\end{array}
$$

Proved progress & preservation

Implemented in Idris

Still to do…

- Task equality
- Pre- and postconditions
- Symbolic execution

IDEAS APPRECIATED!

Institute for Computing and Information Sciences
Radboud University

Thank you

Institute for Computing
and Information Sciences
Radboud University

Summary

- Language & formal semantics
- Proved progress & preservation
- Implemented in Idris
- Essence of task oriented programming

Institute for Computing
and Information Sciences
Radboud University

Grammar

TAKE A Λ-CALCULUS...

$$
\begin{array}{lll}
e ::= & & \text{Expressions} \\
\quad | & \lambda x : \tau.\, e \mid e_1\ e_2 & \text{– abstraction, application} \\
\quad | & x \mid c \mid e_1 \star e_2 & \text{– variable, constant, operation} \\
\quad | & \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 \mid \langle\rangle & \text{– branch, unit} \\
\quad | & \langle e_1, e_2\rangle \mid \mathsf{fst}\, e \mid \mathsf{snd}\, e & \text{– pair, projections} \\
\quad | & \textbf{ref}\, e \mid !e \mid e_1 := e_2 \mid l & \text{– references, location} \\
\quad | & p & \text{– pretask} \\
c ::= & & \text{Constants} \\
\quad | & B \mid I \mid S & \text{– boolean, integer, string}
\end{array}
$$

...EMBED A WORKFLOW LANGUAGE

$$
\begin{array}{lll}
t ::= & & \text{Tasks} \\
\quad | & \square\, v \mid \boxtimes\, \tau \mid \blacksquare\, l & \text{– editors} \\
\quad | & t_1 \blacktriangleright e_2 \mid t_1 \triangleright e_2 & \text{– steps} \\
\quad | & \lightning \mid t_1 \bowtie t_2 & \text{– fail, combination} \\
\quad | & t_1 \blacklozenge t_2 \mid e_1 \lozenge e_2 & \text{– choices}
\end{array}
$$

COMBINATION OF TWO TASKS

RACE BETWEEN TWO TASKS

Summary

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

$$
\begin{array}{lll}
t ::= & & \text{Tasks} \\
\quad | & \square\, v \;\mid\; \boxtimes\, \tau \;\mid\; \blacksquare\, l & \text{– editors} \\
\quad | & t_1 \blacktriangleright e_2 \;\mid\; t_1 \triangleright e_2 & \text{– steps} \\
\quad | & \lightning \;\mid\; t_1 \bowtie t_2 & \text{– fail, combination} \\
\quad | & t_1 \blacklozenge t_2 \;\mid\; e_1 \lozenge e_2 & \text{– choices}
\end{array}
$$

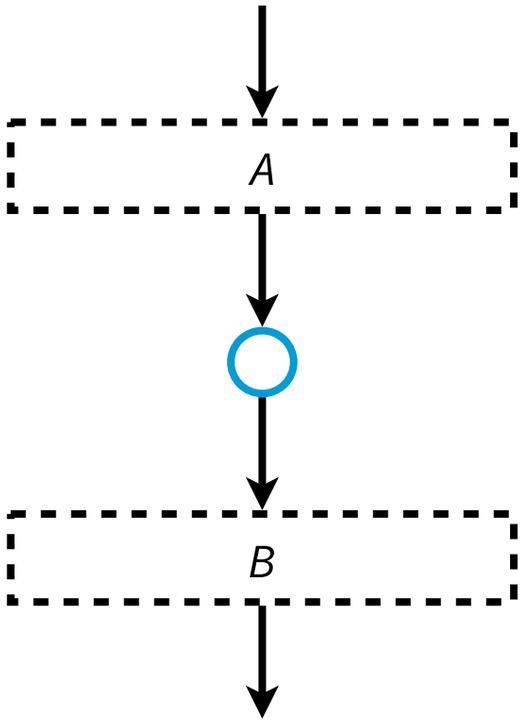
LANGUAGE

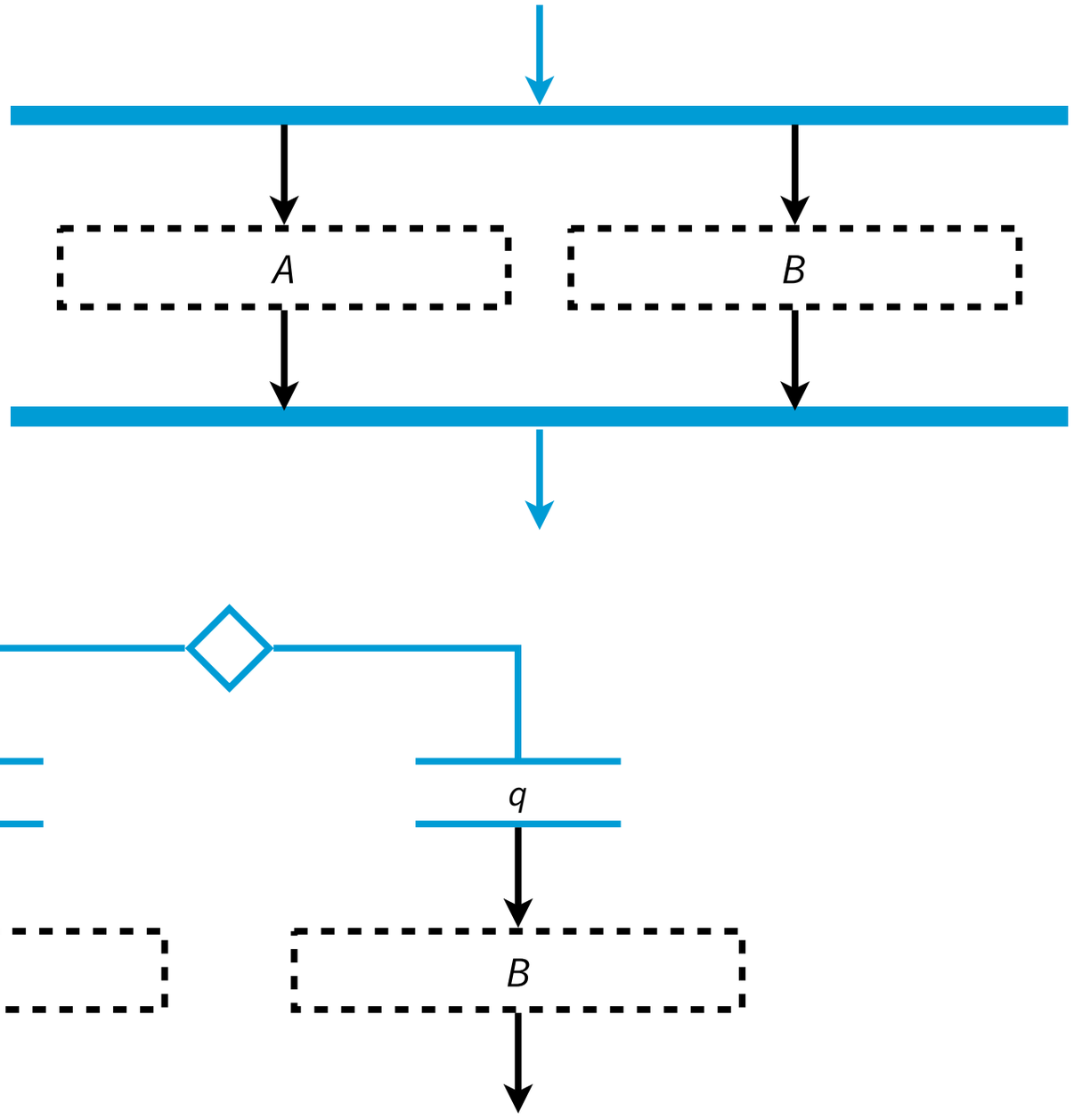
+

FORMAL SEMANTICS

Institute for Computing and Information Sciences
Radboud University

Tasks





l, ⊠⊡⊟▪▫■□■ □■□

l, ⊠⊡⊟▪▫■□■ □■□

■l, ⊠⊡⊟▪▫■□■ □■□

▷▶⏵▷