

Keeping the flow going

Data-flow oriented workflow systems

Annette Bieniusa

joint work with Adriaan Middelkoop(vwd) and Alexander Mattes



„How do you want to invest
your money?“



Financial Advisory Systems

- Financial Advisory
 - Manage estate of clients
 - Giving clients individual advice

Workflows

Dynamic collection of interdependent tasks

- Infrastructure required
 - Enforce regulation and support supervision
 - Automate parts of the processes

User Interface

The screenshot shows a software interface for an 'Einstiegsseite' (Introduction page). The interface is divided into a left sidebar and a main content area. The sidebar contains a 'Dynamic Task Tree' with sections for 'Verknüpft' (linked) and 'Erfassung' (capture). The 'Erfassung' section is expanded, showing a list of tasks: 'Einstiegsseite' (highlighted), 'Formulartyp', 'Grunddaten', 'Anlageziele', 'Finanzielle Verhältnisse', 'Risikoklasse', 'Testperson, A', 'Testperson, B', 'Report WpHG Bogen', 'Report Änderungen', 'Anhänge verwalten', and 'Abschließen'. The main content area contains a 'Concrete (GUI) Task' with the following fields: 'Basisdaten' (Title, Owner, Advisor), 'Zeit' (Start, Duration), and 'Gesprächspartner' (Partner, Additional partners). A 'Suggested next step' button labeled 'Weiter' is located at the bottom right.

Dynamic Task Tree

- Aktivität
 - WpHG-Datenerfassung
 - Letzter Zugriff: 12.01.2015 09:02
 - Inhaber: Testinhaber
- Verknüpft
 - Testinhaber
- Erfassung
 - Einstiegsseite**
 - Formulartyp
 - Grunddaten
 - Anlageziele
 - Finanzielle Verhältnisse
 - Risikoklasse
 - Testperson, A
 - Grunddaten (Kontoinha...
 - Kenntnisse und Erfahru...
 - Testperson, B
 - Grunddaten (Kontoinha...
 - Kenntnisse und Erfahru...
 - Report WpHG Bogen
 - Report Änderungen
 - Anhänge verwalten
 - Abschließen

Concrete (GUI) Task

Dies ist die Einstiegsseite der Aktivität. Erfassen Sie hier die Basisda
Termin und teilnehmenden Personen.

Basisdaten

Betreff _____
Inhaber Testinhaber
Berater * Mustermann, Karl

Zeit

Beginn * 12.01.2015 Zeit 08:58
Gesprächsdauer

- bis 15 Minuten
- ca. 30 Minuten
- mehr als 45 Minuten

Gesprächspartner

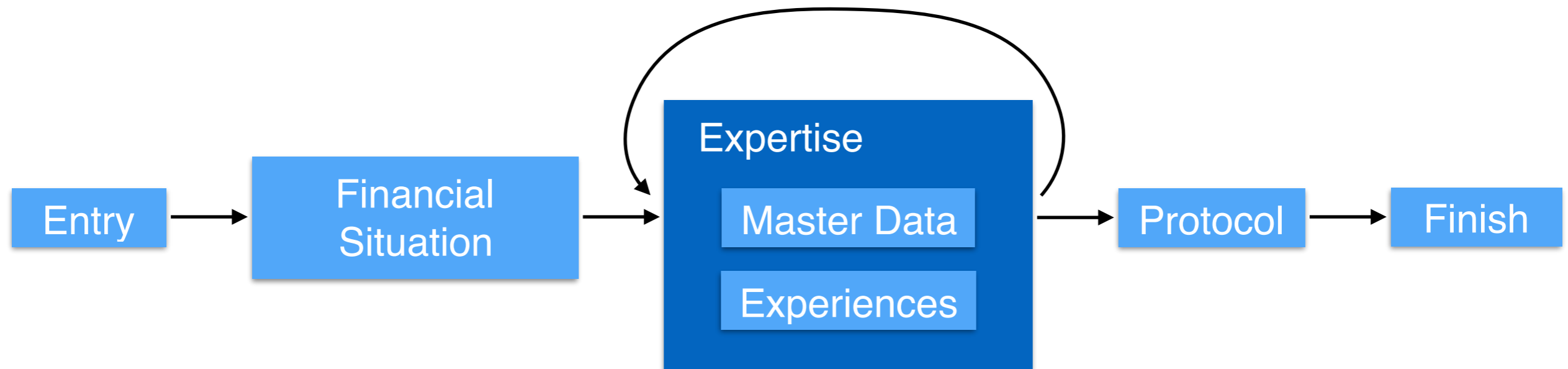
Gesprächspartner * Testperson, A
Weitere Gesprächspartner Testperson, B

Weiter >

Suggested next step

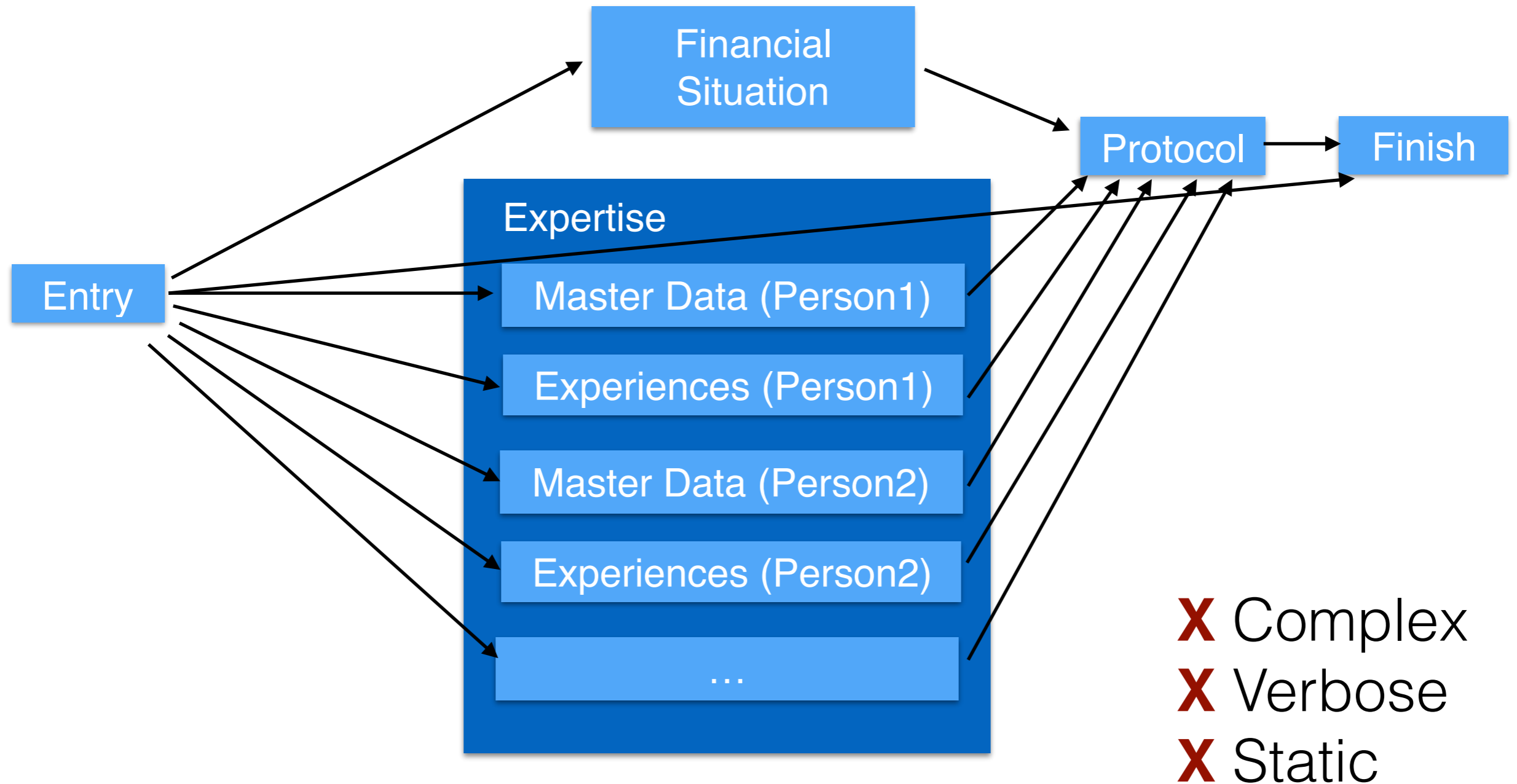
The control-driven approach

Workflow Example: Acquisition of client profile



- X** Fixed task order
- X** Explicit management of shared state

The data-driven approach





IDEA

designed by  freepik.com

Use dynamic, higher-order reactive
programming
to define the (dynamic) dataflow graph

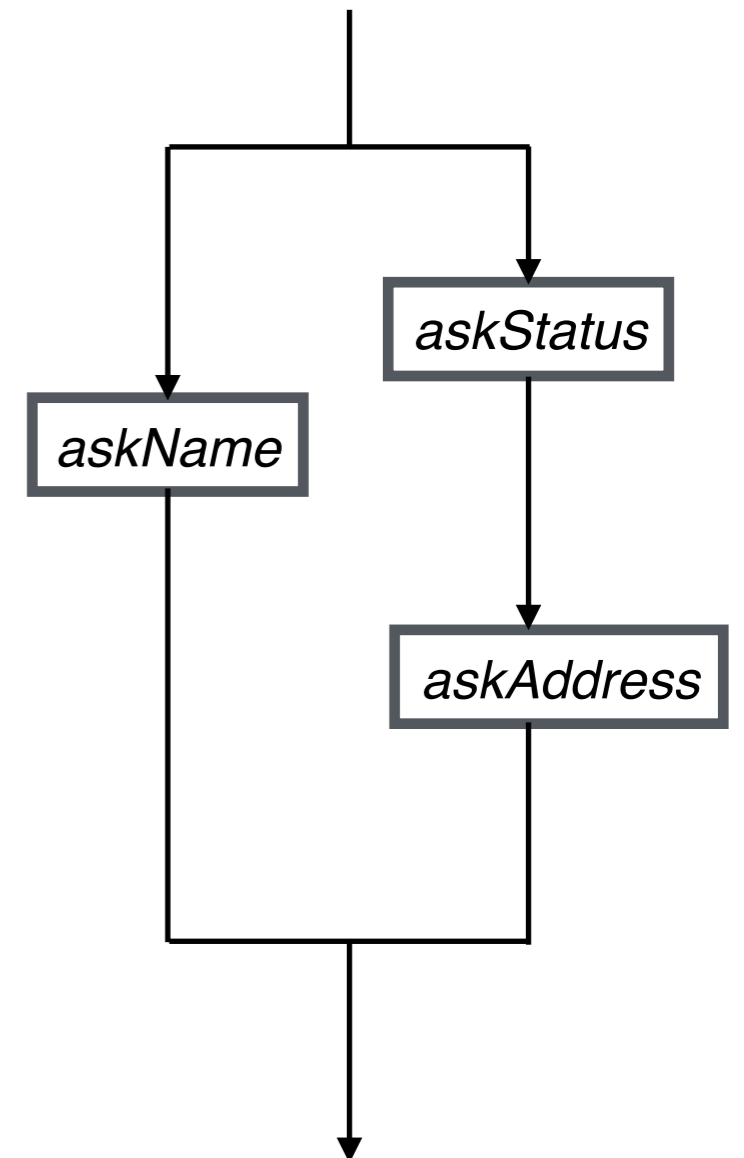
Defining the Tasks

askName :: Task () String
askName = ...

askStatus :: Task () Status — private person or company
askStatus = ...

askAdress = Task Status String
askAdress = ...

workflow :: Task () (String, String)
workflow = ... — combination of Tasks



Types of Tasks

- Need to program the (dynamic) dataflow graph for the workflow
- Idea: Dynamic, higher-order reactive programming

data Task a b where

Pure :: (a -> b) -> Task a b

Impure :: (a -> IO b) -> Task a b

Serial :: Task a x -> Task x b -> Task a b

Parallel :: Task u x -> Task v y -> Task (u,v) (x,y)

here: Task

Tasks as Arrows

- An arrow $a \rightarrow b \rightarrow c$ represents „a computation with input type b delivering something of type c “ [Hughes 00]
- Generalizes the monad concept by introducing dependence on input, e.g.
 - If you are a finance novice, only safe financial products may be proposed
 - Different type of address depending on the status of the advised person

How to construct an Arrow

- Build an arrow out of a function:

$arr :: (Arrow\ a) \Rightarrow (b \rightarrow c) \rightarrow a\ b\ c$

- Arrow composition (sequentiell)

$(>>>) :: (Arrow\ a) \Rightarrow a\ b\ c \rightarrow a\ c\ d \rightarrow a\ b\ d$

- Take an arrow and transform first

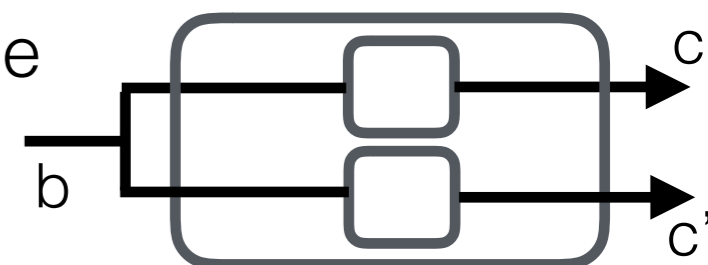
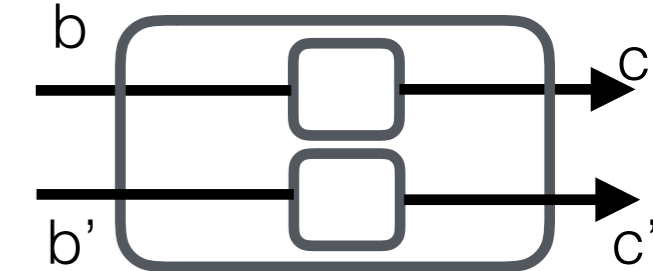
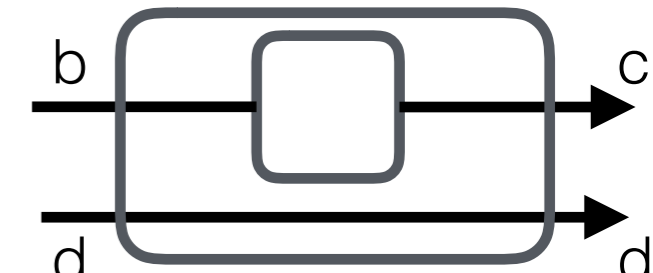
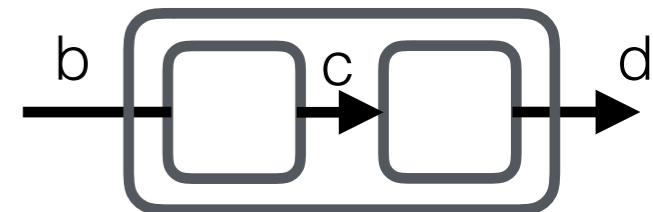
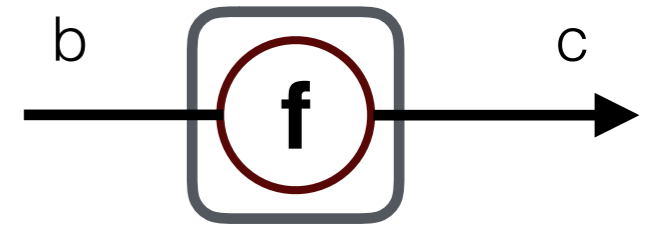
$first :: (Arrow\ a) \Rightarrow a\ b\ c \rightarrow a\ (b, d)\ (c, d)$

- Run one arrow on the first item of the pair and one arrow on the second item of the pair

$(*** :: Arrow\ a \Rightarrow a\ b\ c \rightarrow a\ b'\ c' \rightarrow a\ (b, b')\ (c, c')$

- Combine two arrows by running both on the same value

$(&&&) :: Arrow\ a \Rightarrow a\ b\ c \rightarrow a\ b\ c' \rightarrow a\ b\ (c, c')$



Tasks as Arrows

data Task a b where

Pure :: (a -> b) -> Task a b

Impure :: (a -> IO b) -> Task a b

Serial :: Task a x -> Task x b -> Task a b

Parallel :: Task u x -> Task v y -> Task (u,v) (x,y)

instance Cat.Category Task where

id = Pure id

t1 . t2 = Serial t2 t1

instance Arrow Task where

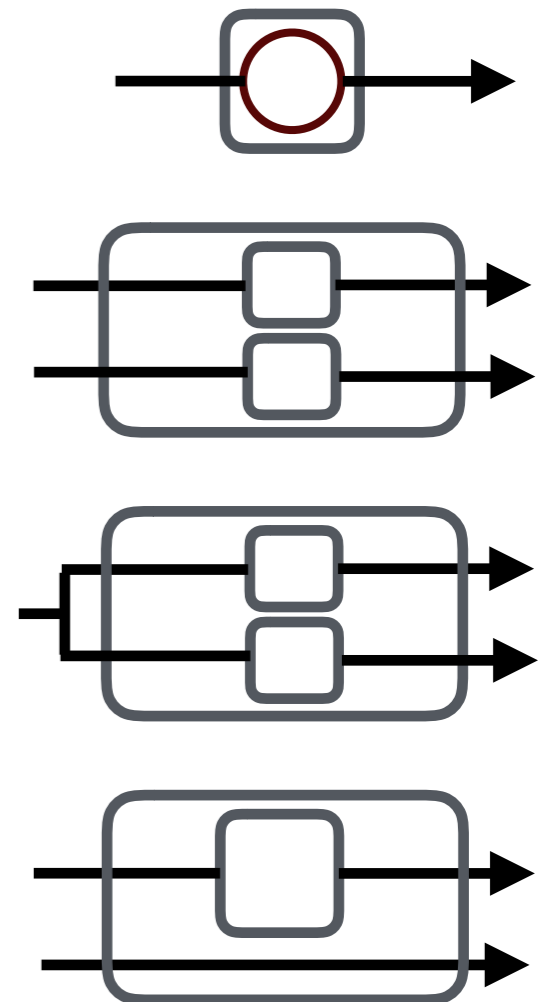
arr = Pure

*t1 *** t2 = Parallel t1 t2*

*t1 &&& t2 = Pure (\a -> (a,a)) >>> t1 *** t2*

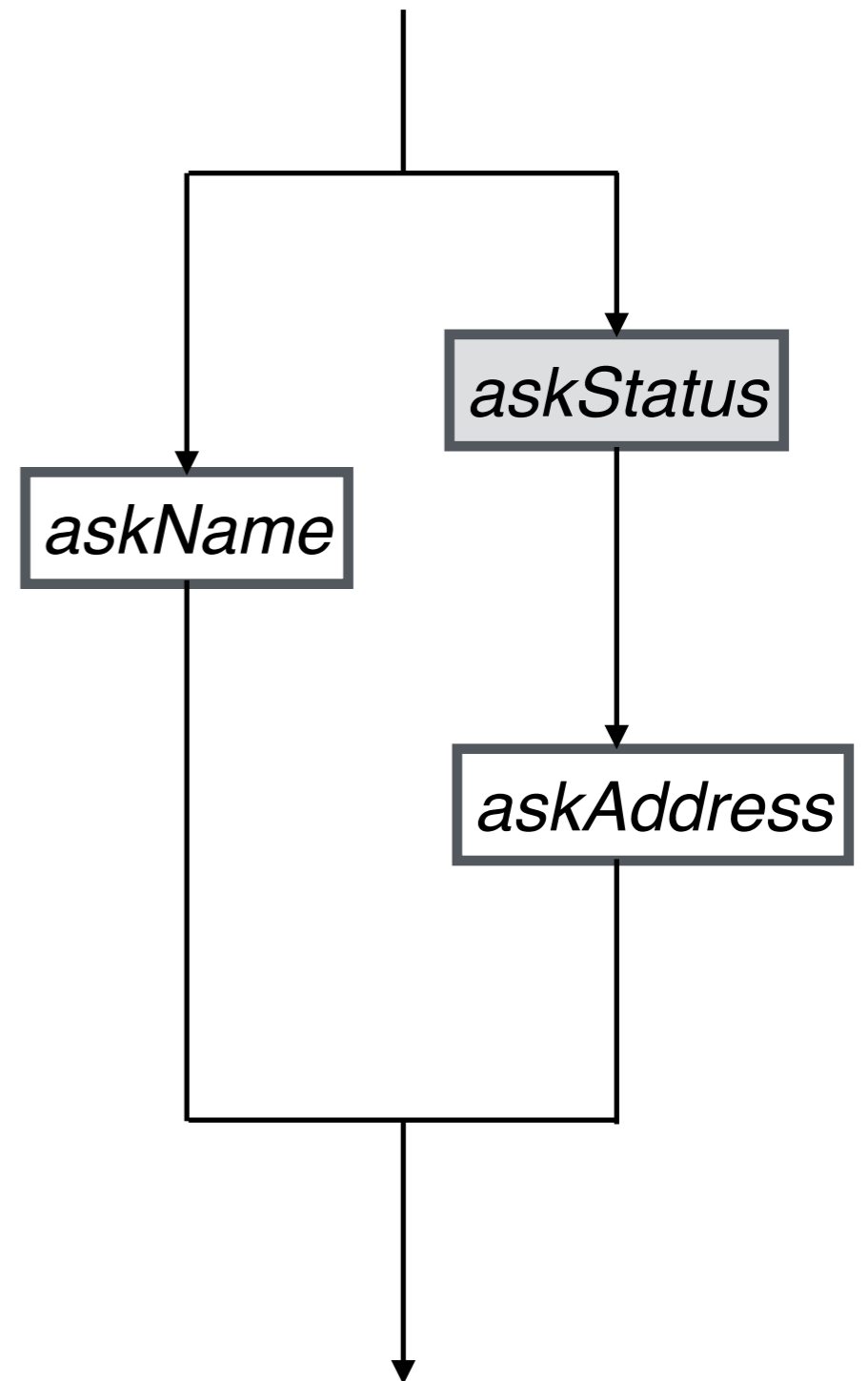
*first t = t *** Cat.id*

*second t = Cat.id *** t*



Keep the data flowing

- Dynamic workflow execution
runTask :: Task a b -> a -> IO b
- Finds the tasks can be processed next
- Allows re-execution of tasks



ArrowTasks

- Haskell library for task-based workflows
- Integrated in Web-Framework (based on HappStack)
- Work in progress:
 - Collaborative workflows
 - Tasks as microservices with workflow API that wraps around a conventional microservice

