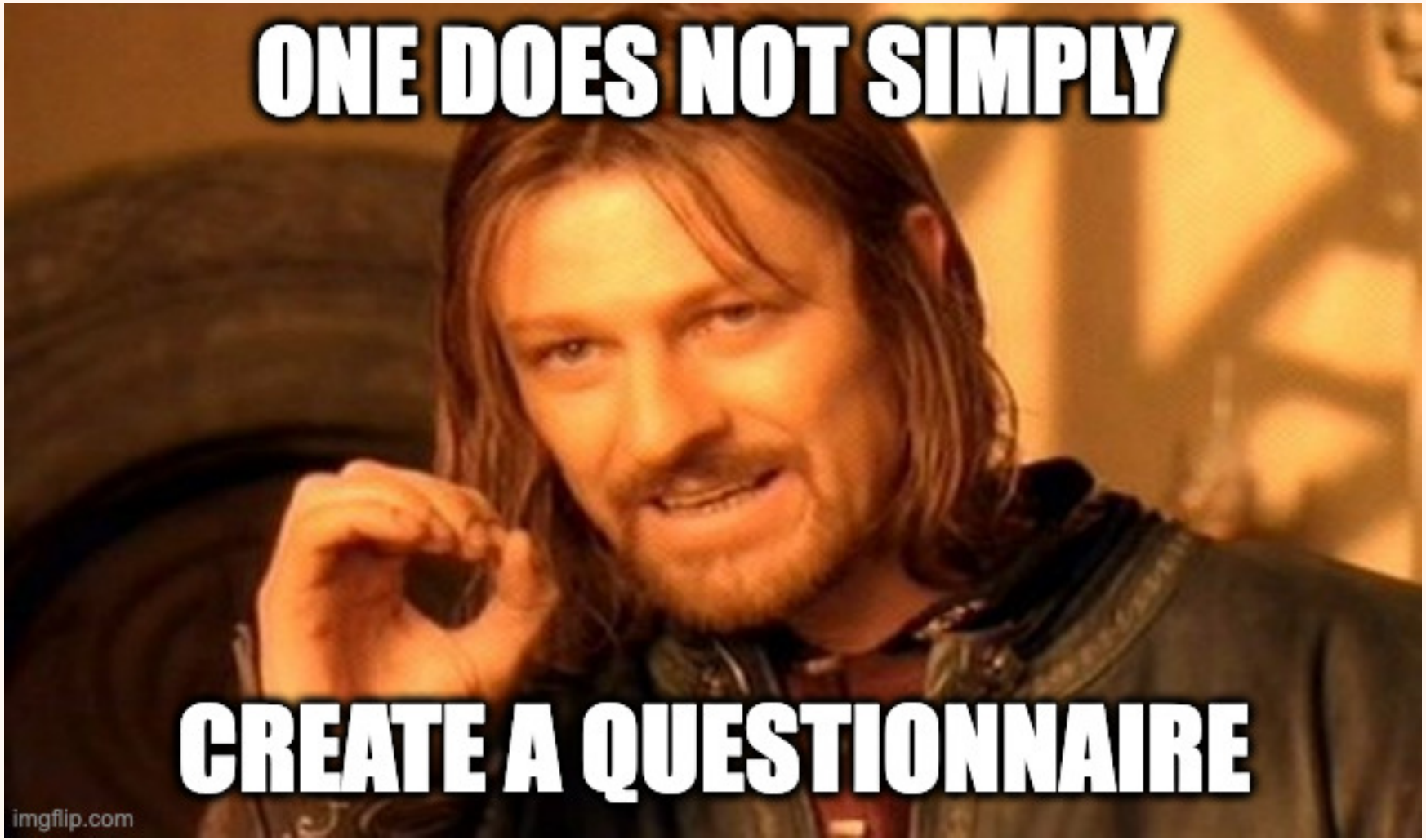
Dependent types are everywhere!

With application to questionnaires


ONE DOES NOT SIMPLY
CREATE A QUESTIONNAIRE
imgflip.com

Dependent types

Dependent types

```
-- 'List' depends on a data type
```

Dependent types

```
-- 'List' depends on a data type
myList : List Int
myList = [1, 2, 3]
```

Dependent types

```
-- 'List' depends on a data type
myList : List Int
myList = [1, 2, 3]

-- Usually: myList[4] -> Exception!
```

Dependent types

```
-- 'List' depends on a data type
myList : List Int
myList = [1, 2, 3]

-- Usually: myList[4] -> Exception!

-- The type of indices in a list
```

Dependent types

```
-- 'List' depends on a data type
myList : List Int
myList = [1, 2, 3]

-- Usually: myList[4] -> Exception!

-- The type of indices in a list
Index : List a -> Type
Index list = Fin (length list) -- {0, 1, ... length list - 1}
```

Dependent types

```
-- 'List' depends on a data type
myList : List Int
myList = [1, 2, 3]

-- Usually: myList[4] -> Exception!

-- The type of indices in a list
Index : List a -> Type
Index list = Fin (length list) -- {0, 1, ... length list - 1}

-- 'Index myList' is the type {0, 1, 2}
```

Dependent types

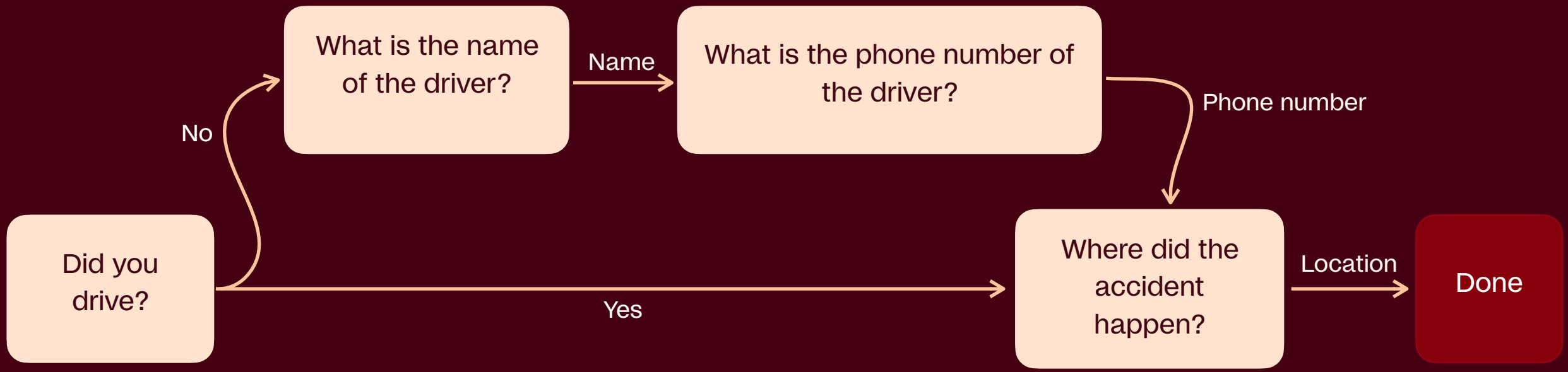
```
-- 'List' depends on a data type
myList : List Int
myList = [1, 2, 3]

-- Usually: myList[4] -> Exception!

-- The type of indices in a list
Index : List a -> Type
Index list = Fin (length list) -- {0, 1, ... length list - 1}

-- 'Index myList' is the type {0, 1, 2}

-- Get an element from the list
-- Cannot get index out of bounds!
```

Dependent types

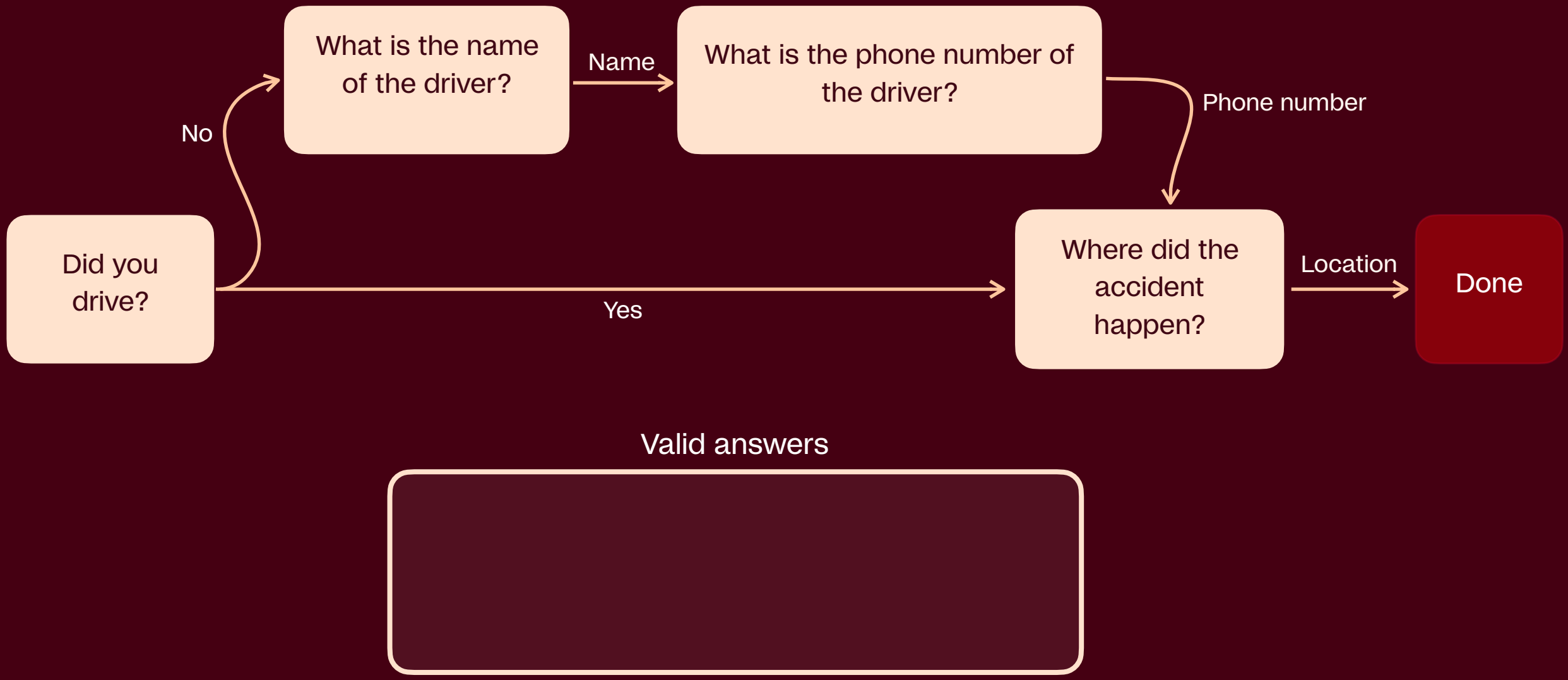
```
-- 'List' depends on a data type
myList : List Int
myList = [1, 2, 3]

-- Usually: myList[4] -> Exception!

-- The type of indices in a list
Index : List a -> Type
Index list = Fin (length list) -- {0, 1, ... length list - 1}

-- 'Index myList' is the type {0, 1, 2}

-- Get an element from the list
-- Cannot get index out of bounds!
safeGet : (list : List a) -> (i : Index list) -> a
safeGet = ...
```


OOP
FP
FP WITH DEPENDENT TYPES
imgflip.com

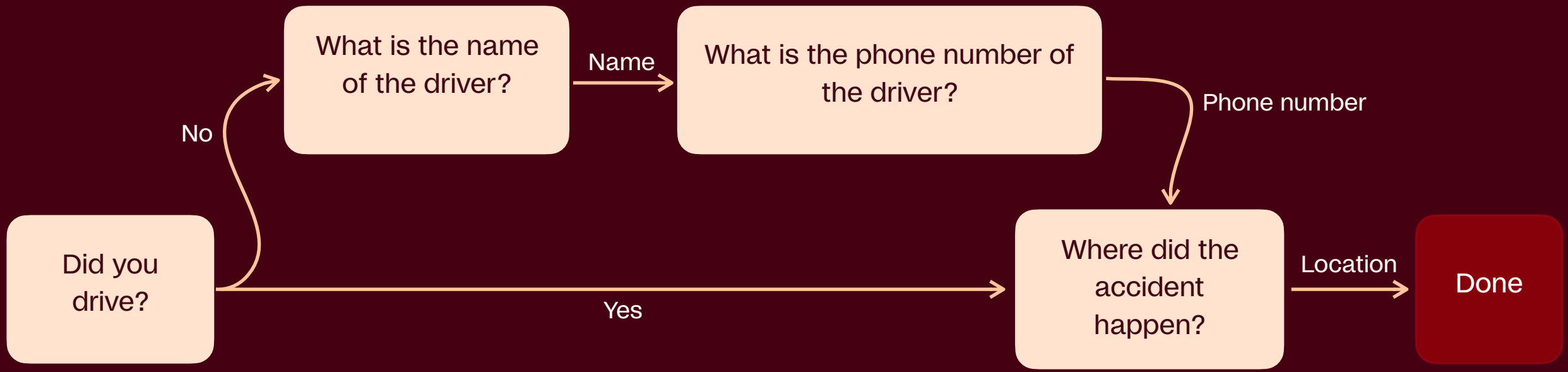
Questionnaires are dependently typed!



Questionnaires are dependently typed!



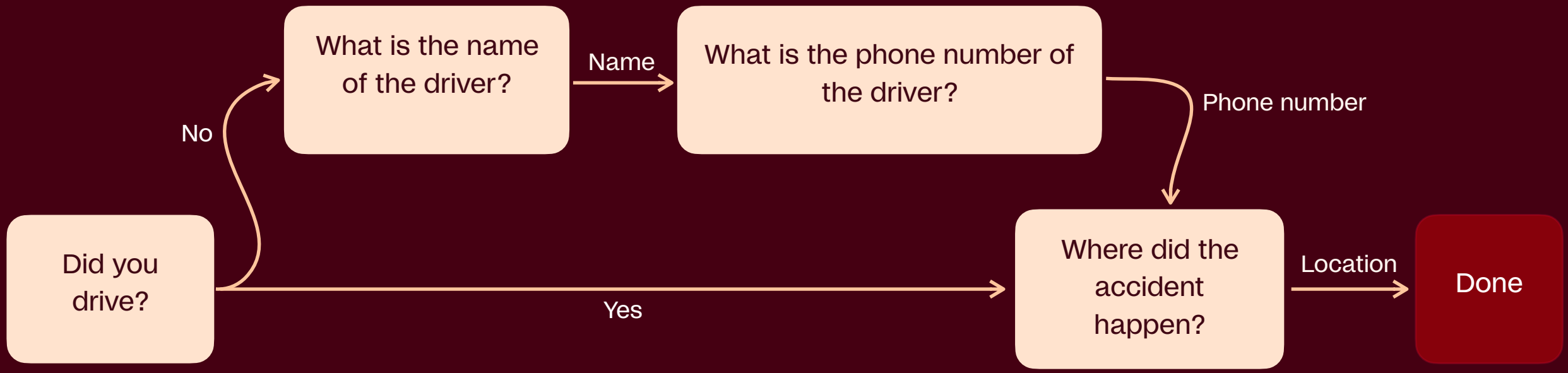
Questionnaires are dependently typed!



Valid answers

Yes → Within Norway

Questionnaires are dependently typed!

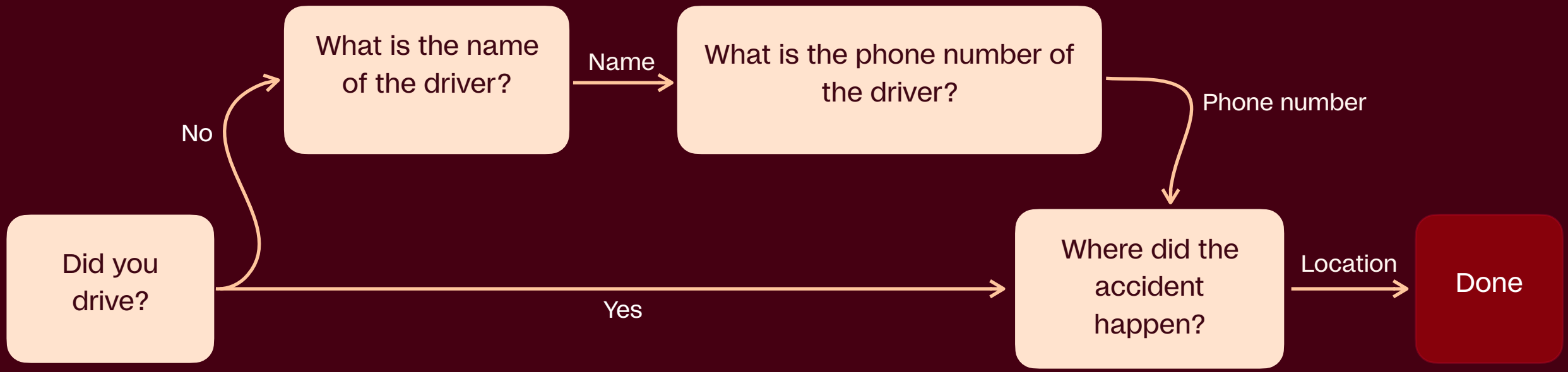


Valid answers

Yes → Within Norway

No → "Hatti Fnatt" → 947 82 456 → Outside Norway

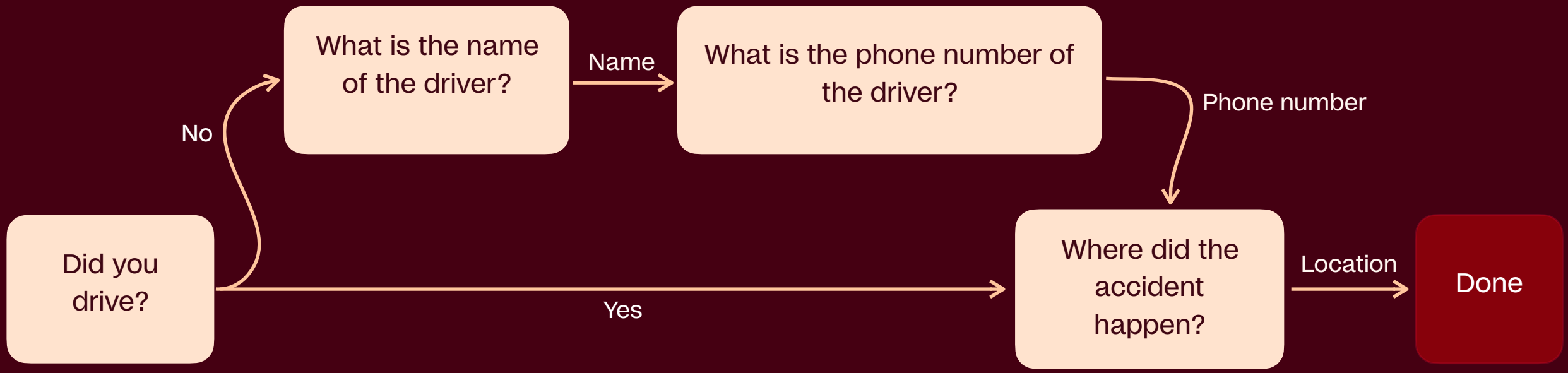
Questionnaires are dependently typed!



Valid answers

Yes → Within Norway
No → "Hatti Fnatt" → 947 82 456 → Outside Norway
No → "Pelle Pang" → 473 62 115 → Within Norway

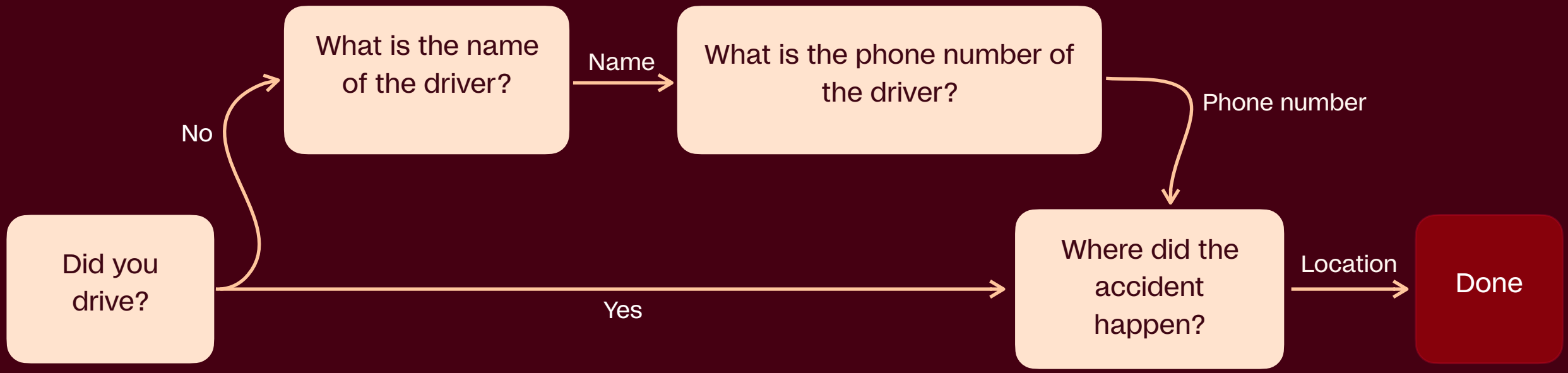
Questionnaires are dependently typed!



Valid answers

Yes → Within Norway
No → "Hatti Fnatt" → 947 82 456 → Outside Norway
No → "Pelle Pang" → 473 62 115 → Within Norway
⋮

Questionnaires are dependently typed!

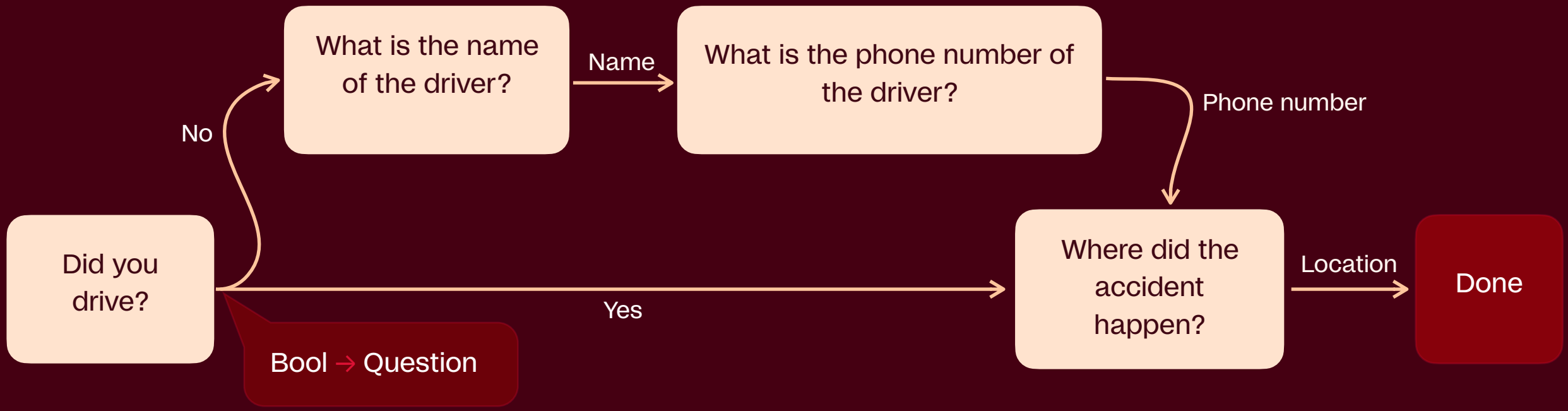


Valid answers

Yes → Within Norway
No → "Hatti Fnatt" → 947 82 456 → Outside Norway
No → "Pelle Pang" → 473 62 115 → Within Norway
⋮

Depends on the graph!

Questionnaires are dependently typed!

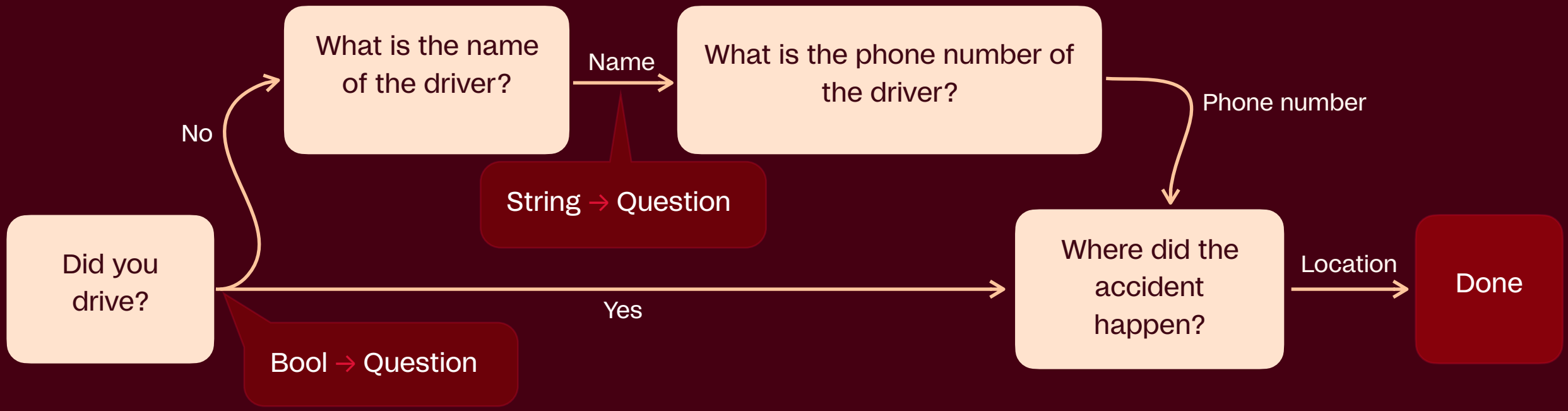


Valid answers

Yes → Within Norway
No → "Hatti Fnatt" → 947 82 456 → Outside Norway
No → "Pelle Pang" → 473 62 115 → Within Norway
⋮

Depends on the graph!

Questionnaires are dependently typed!



Valid answers

Yes → Within Norway
No → "Hatti Fnatt" → 947 82 456 → Outside Norway
No → "Pelle Pang" → 473 62 115 → Within Norway
⋮

Depends on the graph!

Snippets in Idris2

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
           -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
           -> Questionnaire dataType
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
           -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
           -> Questionnaire dataType
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType

-- First question
nextQuestion : Bool -> Questionnaire ValidData
nextQuestion True = Question.Location.question InsuranceHolder
nextQuestion False = Question.Name.question
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType

-- First question
nextQuestion : Bool -> Questionnaire ValidData
nextQuestion True = Question.Location.question InsuranceHolder
nextQuestion False = Question.Name.question
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType

-- First question
nextQuestion : Bool -> Questionnaire ValidData
nextQuestion True = Question.Location.question InsuranceHolder
nextQuestion False = Question.Name.question
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType

-- First question
nextQuestion : Bool -> Questionnaire ValidData
nextQuestion True = Question.Location.question InsuranceHolder
nextQuestion False = Question.Name.question

-- Name question
nextQuestion : Name -> Questionnaire ValidData
nextQuestion name = Question.Phonenumber.question name
```

Snippets in Idris2

```
data Questionnaire : (dataType : Type) -> Type where
  Finished : Finished.Data dataType -> Questionnaire dataType
  Question : (questionData : Question.Data)
          -> (nextQuestion : questionData.AnswerType -> Questionnaire dataType)
          -> Questionnaire dataType

-- First question
nextQuestion : Bool -> Questionnaire ValidData
nextQuestion True = Question.Location.question InsuranceHolder
nextQuestion False = Question.Name.question

-- Name question
nextQuestion : Name -> Questionnaire ValidData
nextQuestion name = Question.Phonenumber.question name
```

Snippets in Idris2

Snippets in Idris2

```
Digit : Type
Digit = Fin 10
```

Snippets in Idris2

```
Digit : Type
Digit = Fin 10

IsValidNumber : Vect 8 Digit -> Type
IsValidNumber number = Either (head number = 4) (head number = 9)
```

Snippets in Idris2

```
Digit : Type
Digit = Fin 10

IsValidNumber : Vect 8 Digit -> Type
IsValidNumber number = Either (head number = 4) (head number = 9)

MobilePhoneNumber : Type
MobilePhoneNumber = (number : Vect 8 Digit ** IsValidNumber number)
```

Snippets in Idris2

```
Digit : Type
Digit = Fin 10

IsValidNumber : Vect 8 Digit -> Type
IsValidNumber number = Either (head number = 4) (head number = 9)

MobilePhoneNumber : Type
MobilePhoneNumber = (number : Vect 8 Digit ** IsValidNumber number)

Event : UI.State questionnaire -> Type
Event Init = ()
Event (AtQuestion (Finished finishedData ** _) _) = Void
Event (AtQuestion (Question questionData nextQuestion ** _) state) =
  Either (questionData.Event state) questionData.AnswerType
```

Snippets in Idris2

```
Digit : Type
Digit = Fin 10

IsValidNumber : Vect 8 Digit -> Type
IsValidNumber number = Either (head number = 4) (head number = 9)

MobilePhoneNumber : Type
MobilePhoneNumber = (number : Vect 8 Digit ** IsValidNumber number)

Event : UI.State questionnaire -> Type
Event Init = ()
Event (AtQuestion (Finished finishedData ** _) _) = Void
Event (AtQuestion (Question questionData nextQuestion ** _) state) =
  Either (questionData.Event state) questionData.AnswerType
```

Snippets in Idris2

```
Digit : Type
Digit = Fin 10

IsValidNumber : Vect 8 Digit -> Type
IsValidNumber number = Either (head number = 4) (head number = 9)

MobilePhoneNumber : Type
MobilePhoneNumber = (number : Vect 8 Digit ** IsValidNumber number)

Event : UI.State questionnaire -> Type
Event Init = ()
Event (AtQuestion (Finished finishedData ** _) _) = Void
Event (AtQuestion (Question questionData nextQuestion ** _) state) =
  Either (questionData.Event state) questionData.AnswerType
```

Snippets in Idris2

```
Digit : Type
Digit = Fin 10

IsValidNumber : Vect 8 Digit -> Type
IsValidNumber number = Either (head number = 4) (head number = 9)

MobilePhoneNumber : Type
MobilePhoneNumber = (number : Vect 8 Digit ** IsValidNumber number)

Event : UI.State questionnaire -> Type
Event Init = ()
Event (AtQuestion (Finished finishedData ** _) _) = Void
Event (AtQuestion (Question questionData nextQuestion ** _) state) =
  Either (questionData.Event state) questionData.AnswerType
```

Snippets in Idris2

```
Digit : Type
Digit = Fin 10

IsValidNumber : Vect 8 Digit -> Type
IsValidNumber number = Either (head number = 4) (head number = 9)

MobilePhoneNumber : Type
MobilePhoneNumber = (number : Vect 8 Digit ** IsValidNumber number)

Event : UI.State questionnaire -> Type
Event Init = ()
Event (AtQuestion (Finished finishedData ** _) _) = Void
Event (AtQuestion (Question questionData nextQuestion ** _) state) =
  Either (questionData.Event state) questionData.AnswerType
```

Demo

Takeaway

Takeaway

- Dependent types are everywhere!

Takeaway

- Dependent types are everywhere!
- Expressive but complex.

Takeaway

- Dependent types are everywhere!
- Expressive but complex.
- Coming soon to a Haskell near you.

Takeaway

- Dependent types are everywhere!
- Expressive but complex.
- Coming soon to a Haskell near you.
- Try it yourself!

Takeaway

- Dependent types are everywhere!
- Expressive but complex.
- Coming soon to a Haskell near you.
- Try it yourself!
    - Idris2

Takeaway

- Dependent types are everywhere!
- Expressive but complex.
- Coming soon to a Haskell near you.
- Try it yourself!
  - Idris2
  - Lean4

Takeaway

- Dependent types are everywhere!
- Expressive but complex.
- Coming soon to a Haskell near you.
- Try it yourself!
  - Idris2
  - Lean4
  - Agda

Takeaway

- Dependent types are everywhere!
- Expressive but complex.
- Coming soon to a Haskell near you.
- Try it yourself!
  - Idris2
  - Lean4
  - Agda
  - Rocq