



Hoople+:

Program Synthesis by Type-Guided Abstraction Refinement

Zheng Guo, Michael James, David Justo, Jiaxiao Zhou, Ziteng Wang, Ranjit Jhala, Nadia Polikarpova

University of California San Diego

Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: **Type-Guided Abstraction Refinement**

Abstraction

Refinement

Evaluation

04 Hoogle+: More Features

Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: Type-Guided Abstraction Refinement

Abstraction

Refinement

Evaluation

04 Hoogle+: More Features

Example

Default value	List of optional values	Desired result
'a'	[Nothing, Just 'b', Nothing, Just 'c', Just 'd']	'b'
0	[Nothing, Nothing, Nothing, Nothing]	0
d: a	xs: [Maybe a]	a

Example

Default value	List of optional values	Desired result
'a'	[Nothing , Just 'b' , Nothing , Just 'c', Just 'd']	'b'
0	[Nothing , Nothing , Nothing , Nothing]	0

$d: a \longrightarrow xs: [Maybe\ a] \longrightarrow a$

Component-Based Synthesis

Example

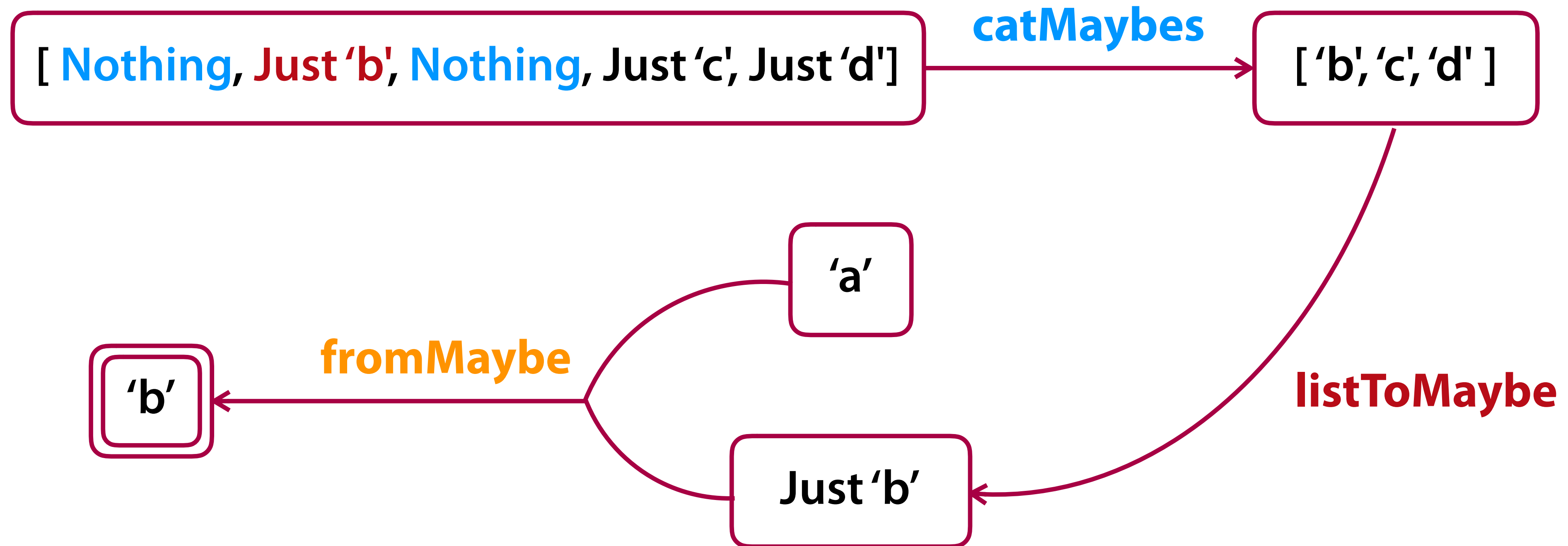
The screenshot shows a web browser window with the URL `hoogle.haskell.org/?scope=set%3Ahaskell-platform`. The page features the Hoogle logo and a search bar containing the text `set:haskell-platform`. Below the search bar, the results are displayed under the heading `set:haskell-platform`. On the left side, there are filter options for `is:exact` and `is:package`. The search results list several packages:

- package base**: Basic libraries. This package contains the Standard Haskell Prelude and its support libraries, and a large collection of useful...
- package bytestring**: Fast, compact, strict and lazy byte strings with a list interface. An efficient compact, immutable byte string type (both strict and lazy), with a list interface. It is suitable for high performance use, both in terms of large data quantities, or high speed require...
- package containers**: Assorted concrete container types. This package contains efficient general-purpose implementations of various immutable containers. The package provides with examples of common operations see the containers introduction. The declared cost of each operation...
- package text**: An efficient packed Unicode text type. An efficient packed, immutable Unicode text type (both strict and lazy), with a powerful list interface. This package provides text processing capabilities that are optimized for performance critical...

A red rectangular box highlights the `package text` entry and its description.

Example

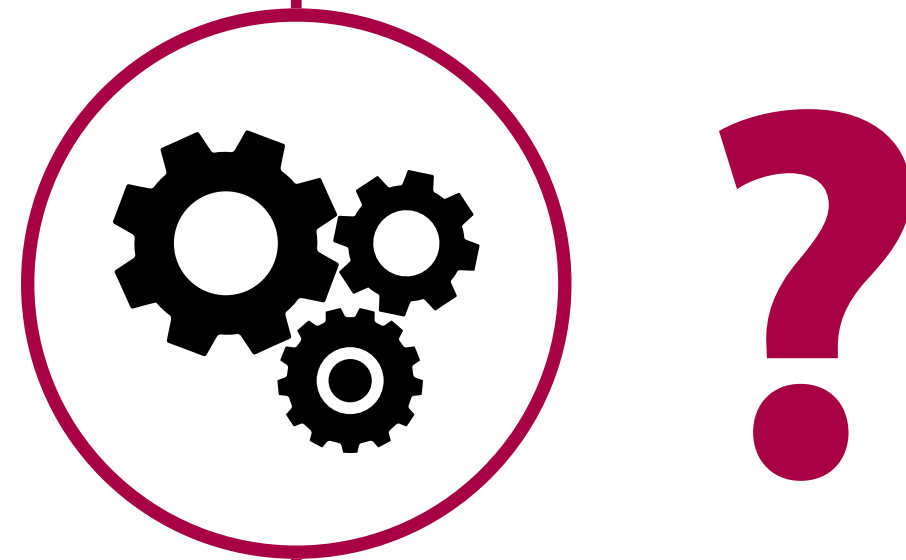
Solution: $\lambda d \text{ xs} \rightarrow \text{fromMaybe } d (\text{listToMaybe } (\text{catMaybes } \text{xs}))$



Component-Based Synthesis

Example

`d: Int -> xs: [Maybe Int] -> Int`



Solution: `\d xs -> fromMaybe d (listToMaybe (catMaybes xs))`

Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: Type-Guided Abstraction Refinement

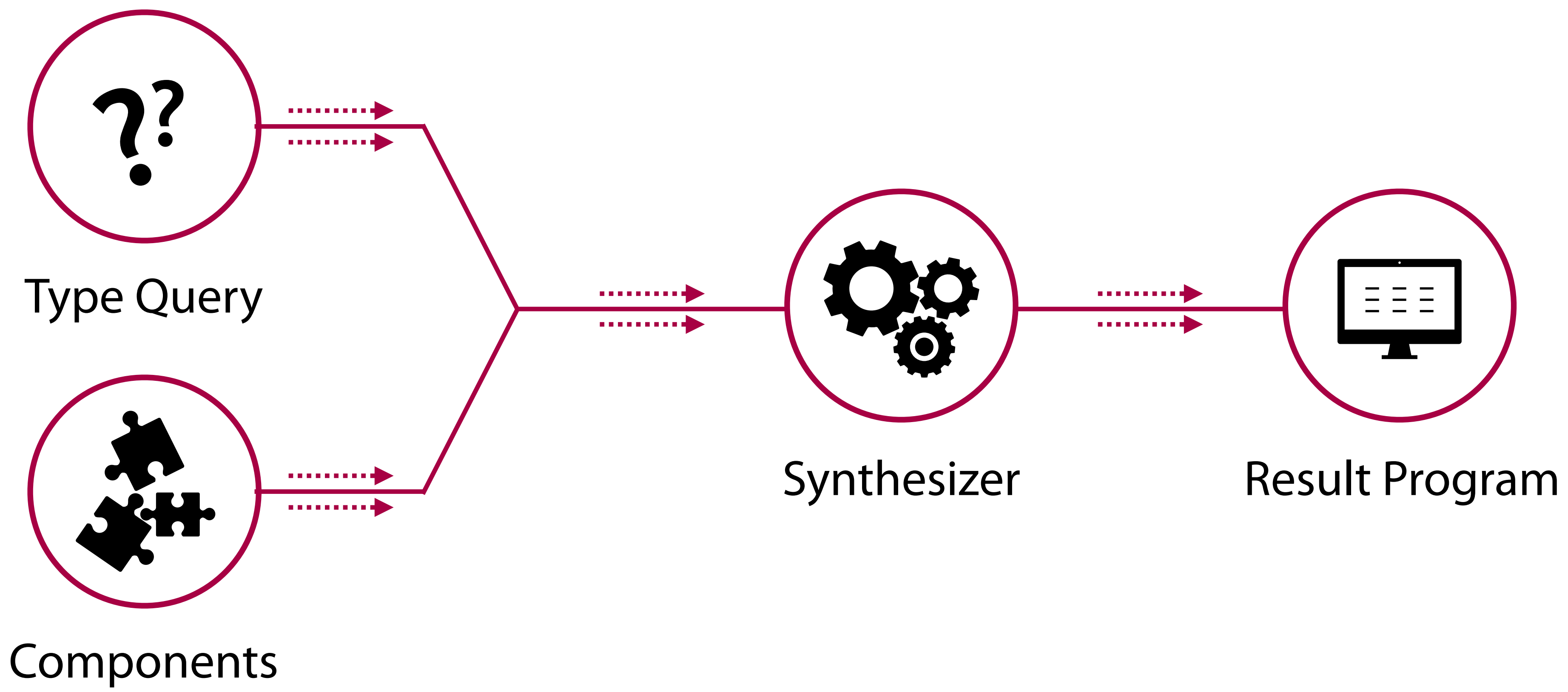
Abstraction

Refinement

Evaluation

04 Hoogle+: More Features





Component-Based Synthesis
Synthesis to the rescue



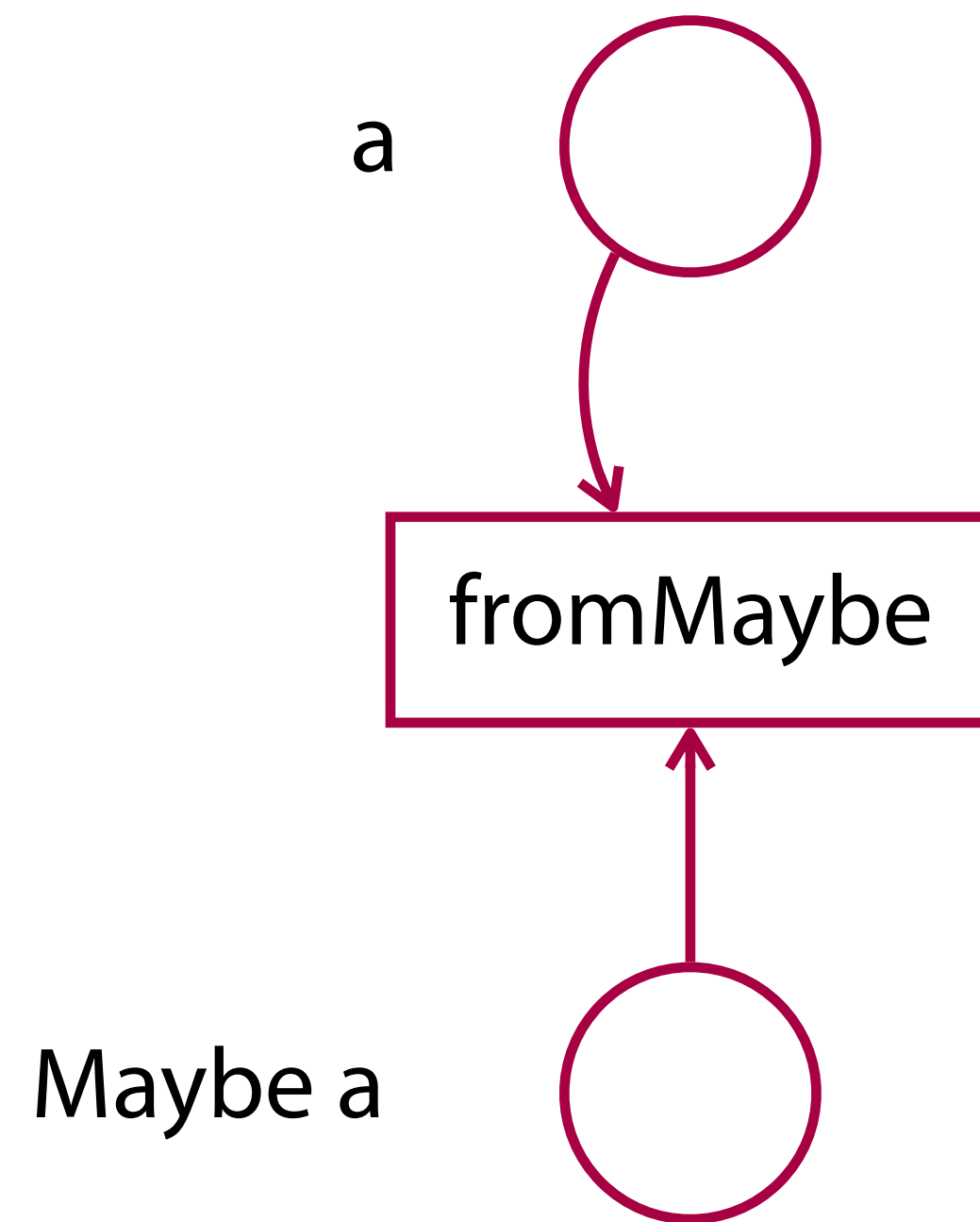
Previous Solution
Petri net-Based Search

fromMaybe :: a -> Maybe a -> a

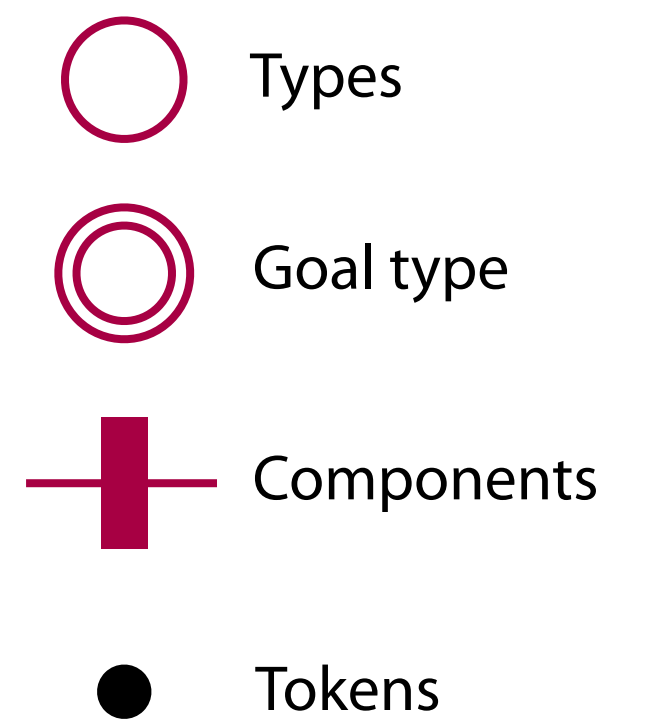
fromMaybe

-  Types
-  Goal type
-  Components
-  Tokens

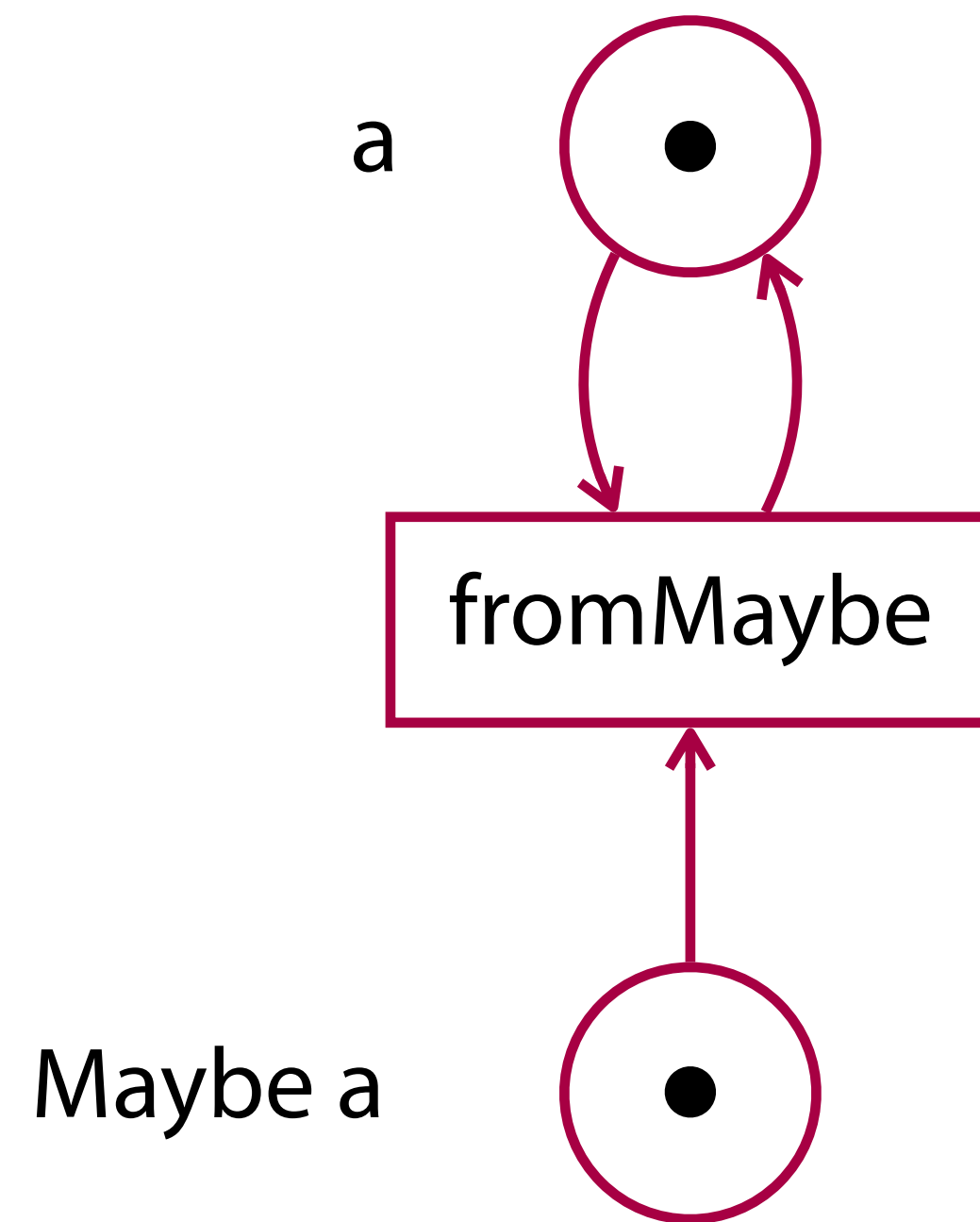
Previous Solution
Petri net-Based Search



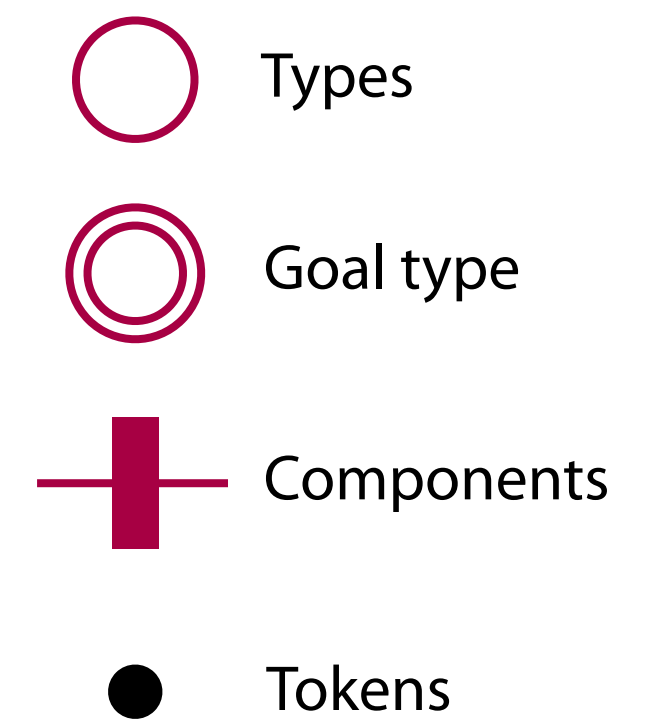
`fromMaybe :: a -> Maybe a -> a`



Previous Solution
Petri net-Based Search



`fromMaybe :: a -> Maybe a -> a`

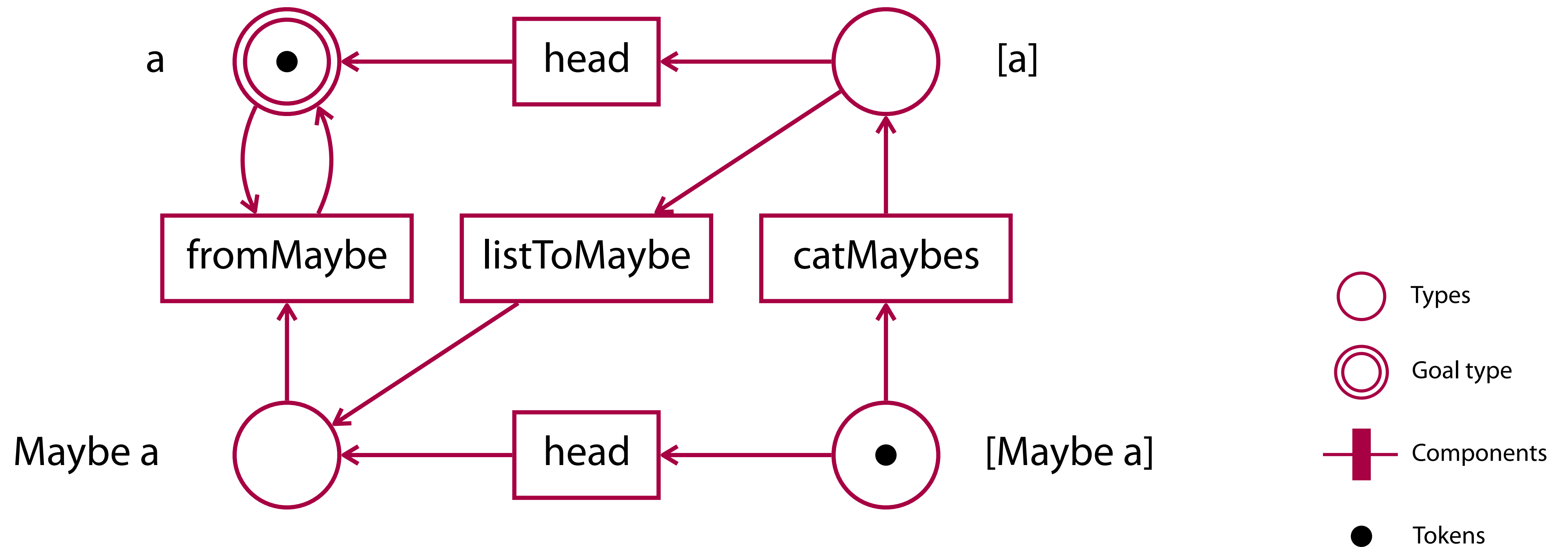


Previous Solution
Petri net-Based Search

Feng et al. POPL '17

Type Query

d: a -> xs: [Maybe a] -> a

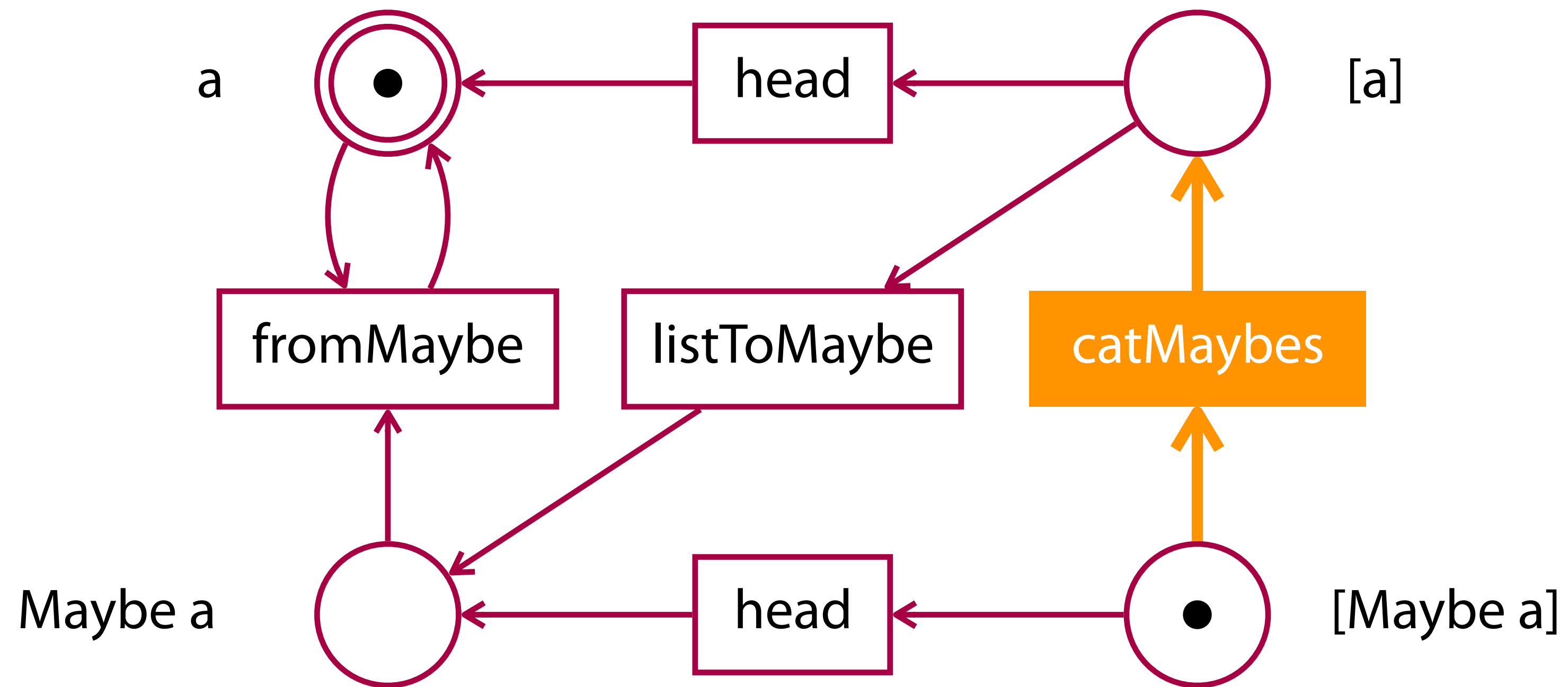


Previous Solution
Petri net-Based Search

Feng et al. POPL '17

Type Query

d: a -> xs: [Maybe a] -> a



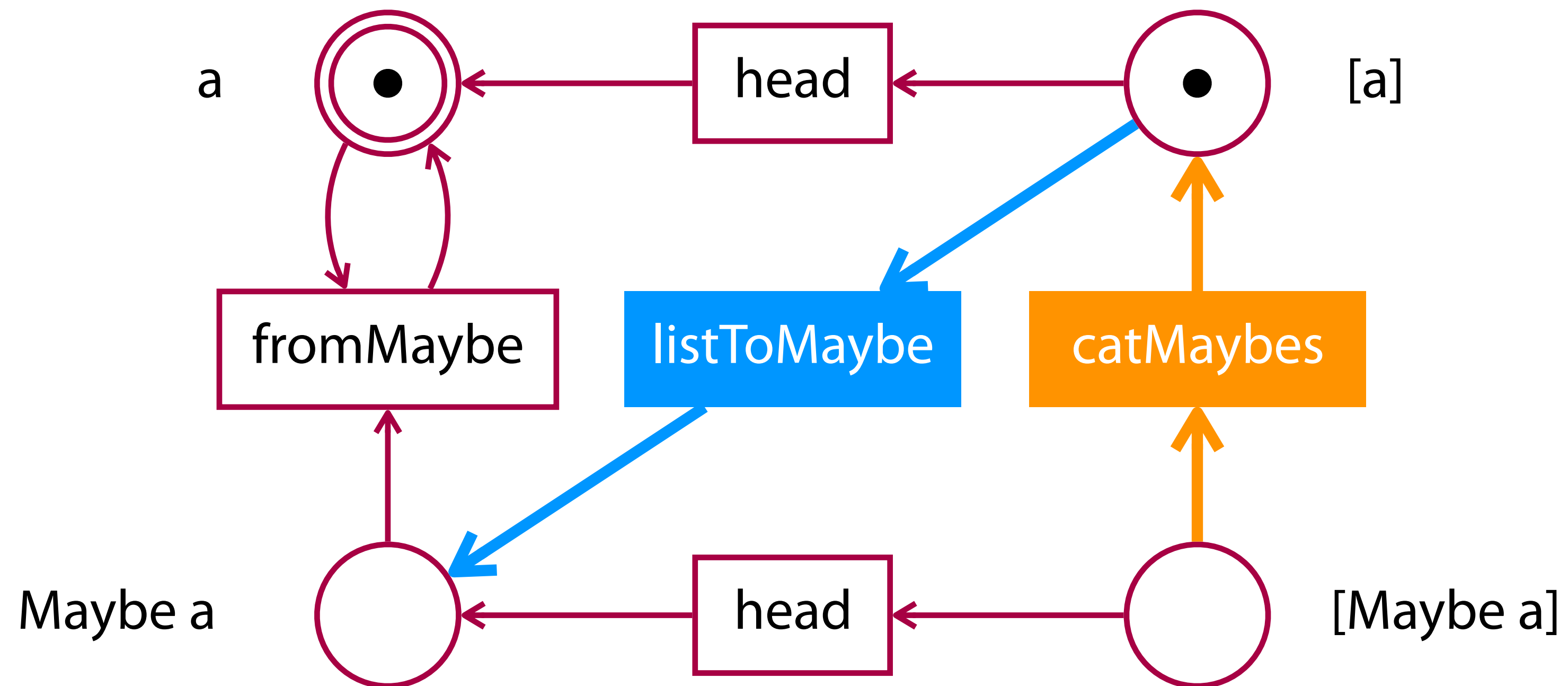
Previous Solution
Petri net-Based Search

Feng et al. POPL '17

Type Query

d: a -> xs: [Maybe a] -> a

catMaybes xs



Previous Solution

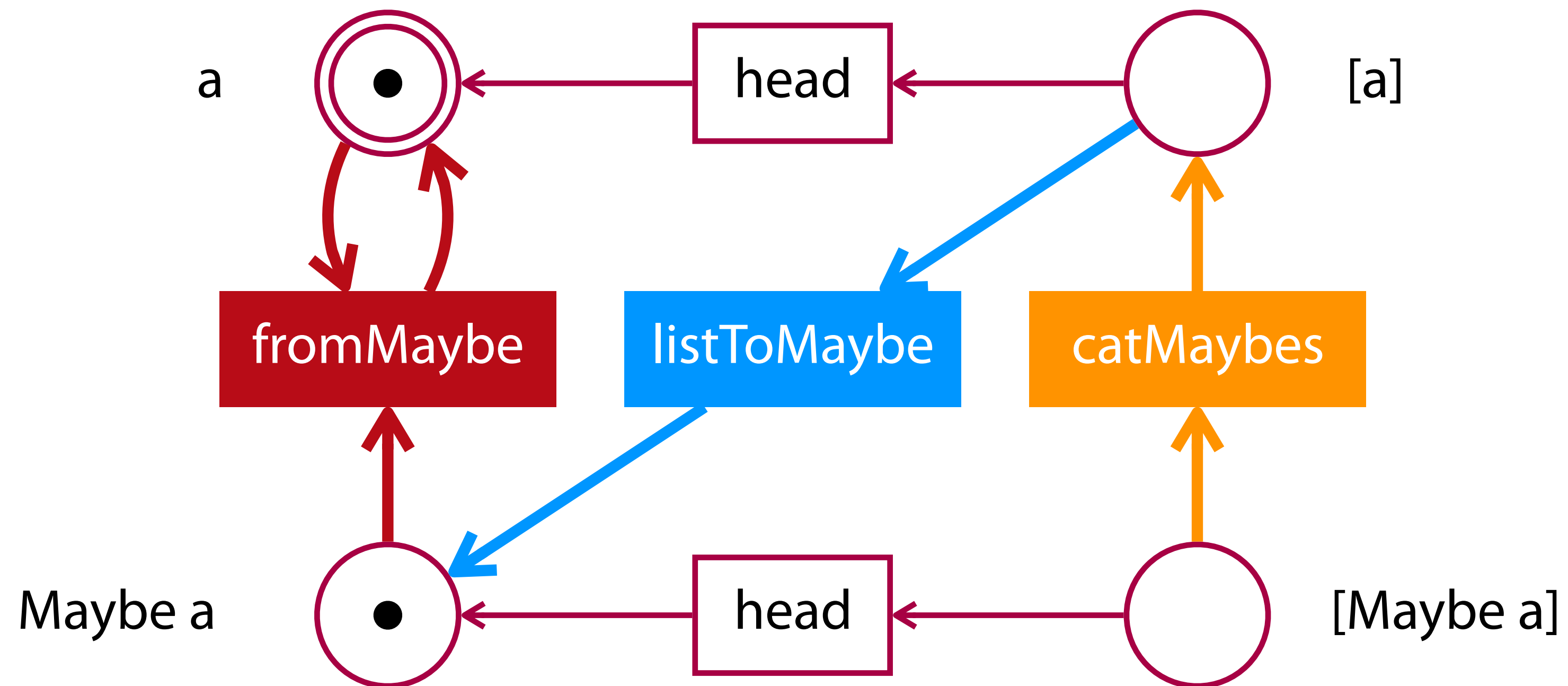
Petri net-Based Search

Feng et al. POPL '17

Type Query

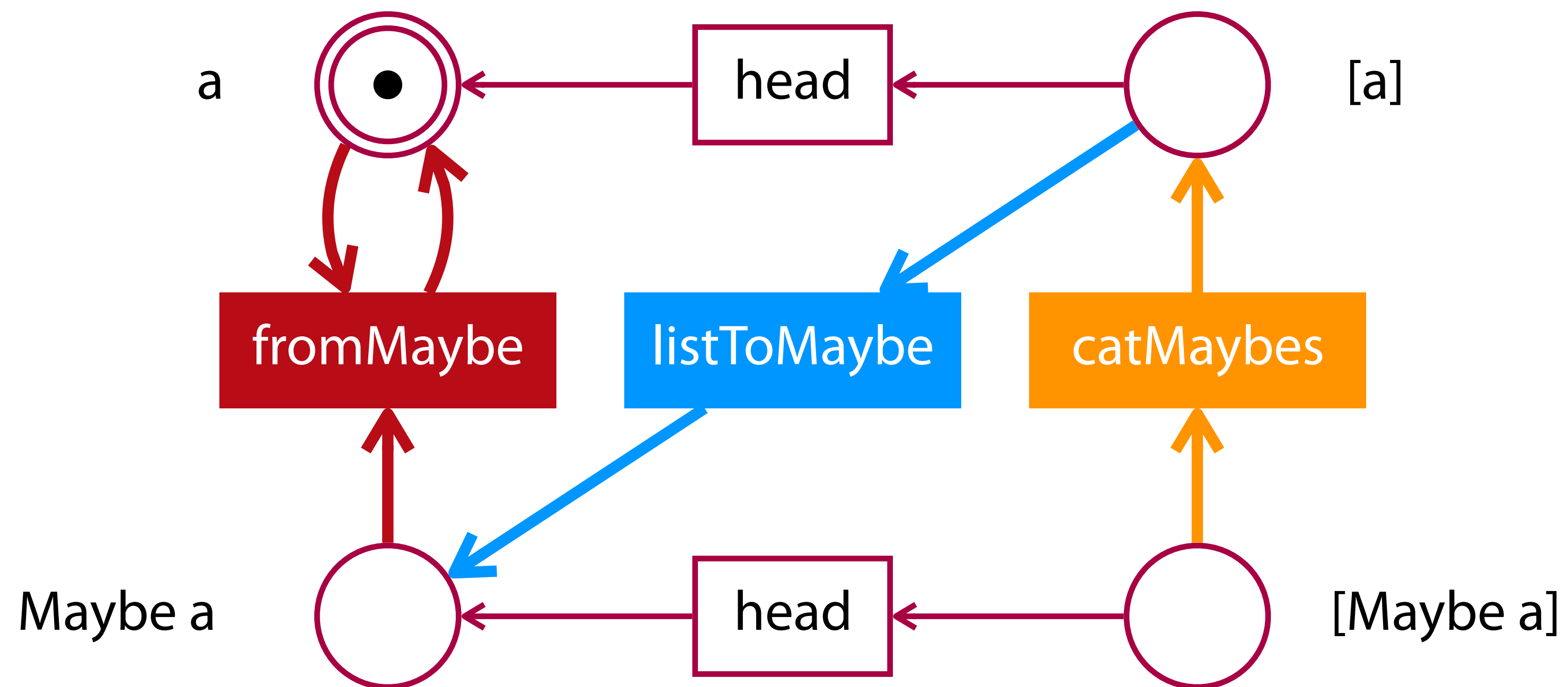
$d: a \rightarrow xs: [Maybe a] \rightarrow a$

listToMaybe (**catMaybes** xs)



Previous Solution
Petri net-Based Search

SOLUTION: $\backslash d \ xs \rightarrow$ **fromMaybe** d (**listToMaybe** (**catMaybes** xs))



Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: Type-Guided Abstraction Refinement

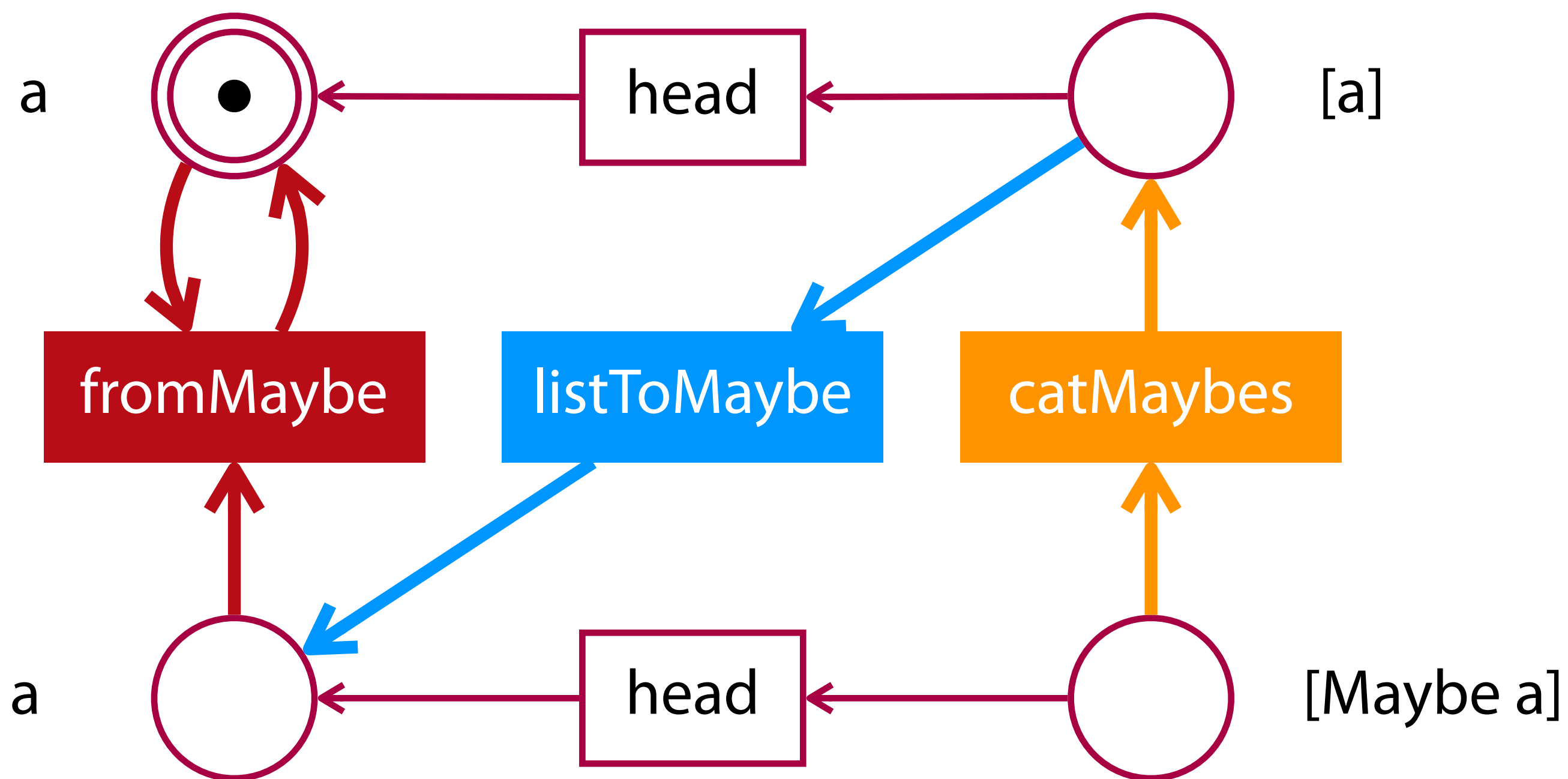
Abstraction

Refinement

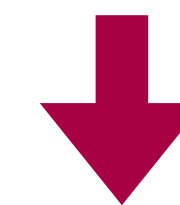
Evaluation

04 Hoogle+: More Features

Challenge
Polymorphic components



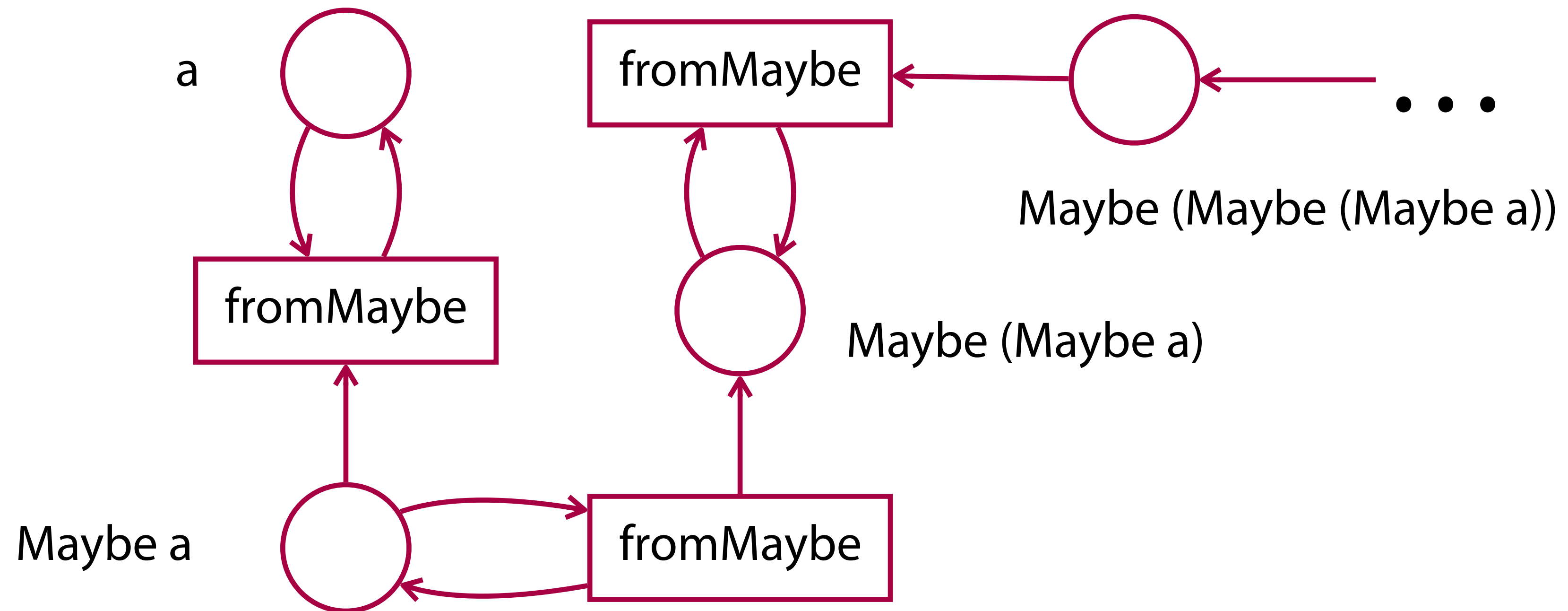
fromMaybe :: a -> Maybe a -> a



fromMaybe :: $\forall \alpha. \alpha \rightarrow \text{Maybe } \alpha \rightarrow \alpha$

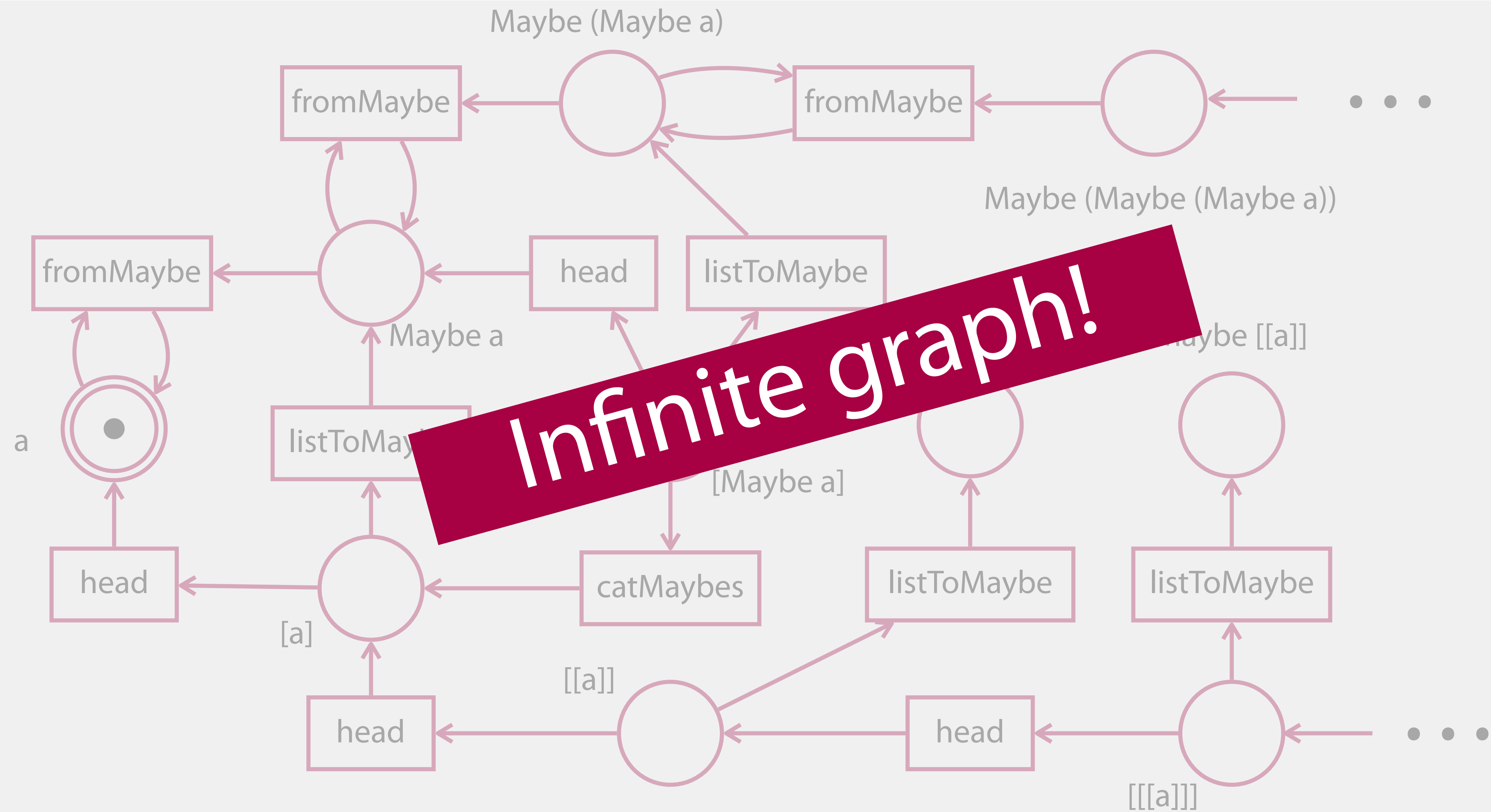
Challenge
Polymorphic components

fromMaybe :: $\forall \alpha. \alpha \rightarrow \text{Maybe } \alpha \rightarrow \alpha$



Challenge

Polymorphic components



Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: **Type-Guided Abstraction Refinement**

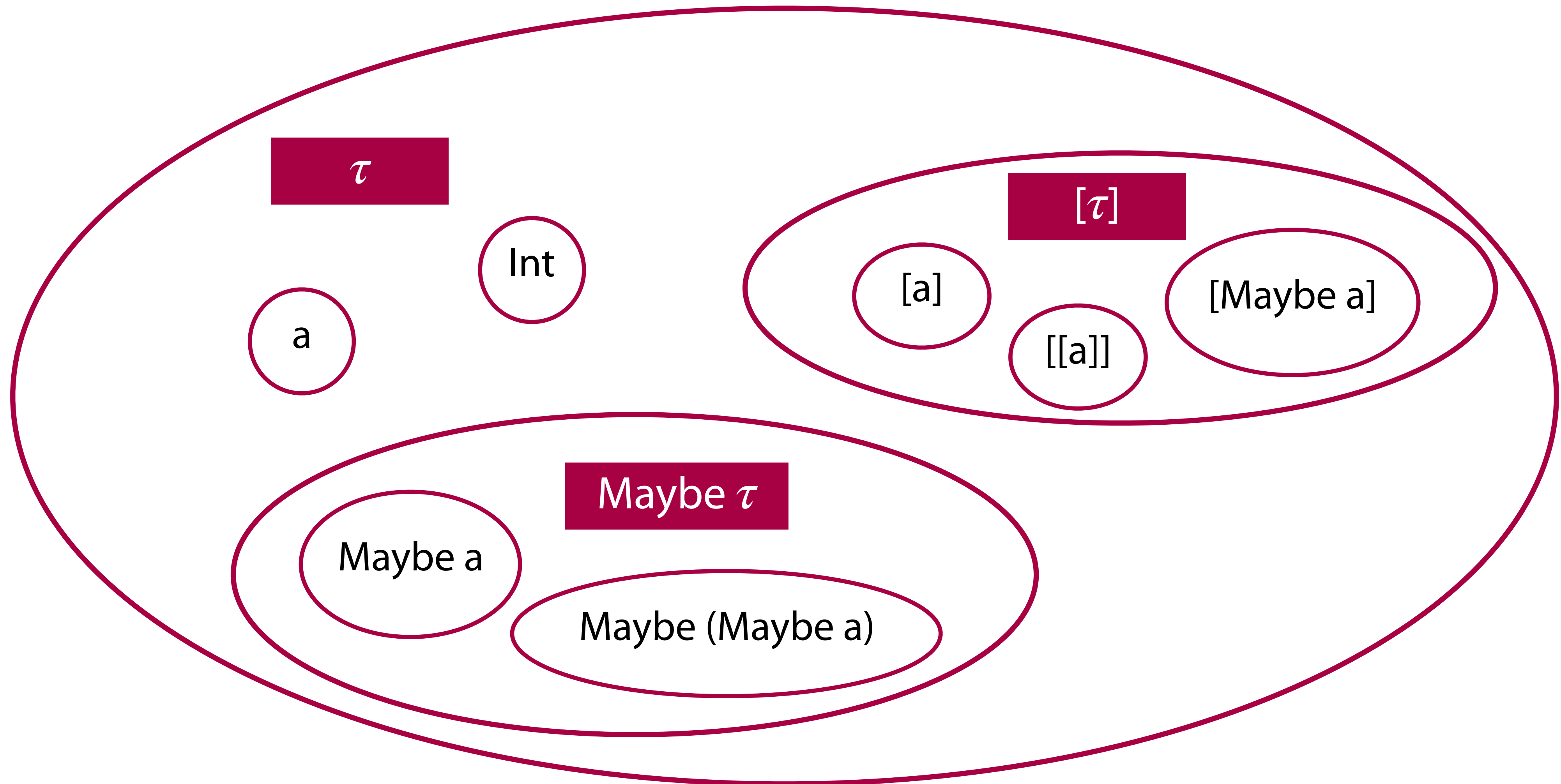
Abstraction

Refinement

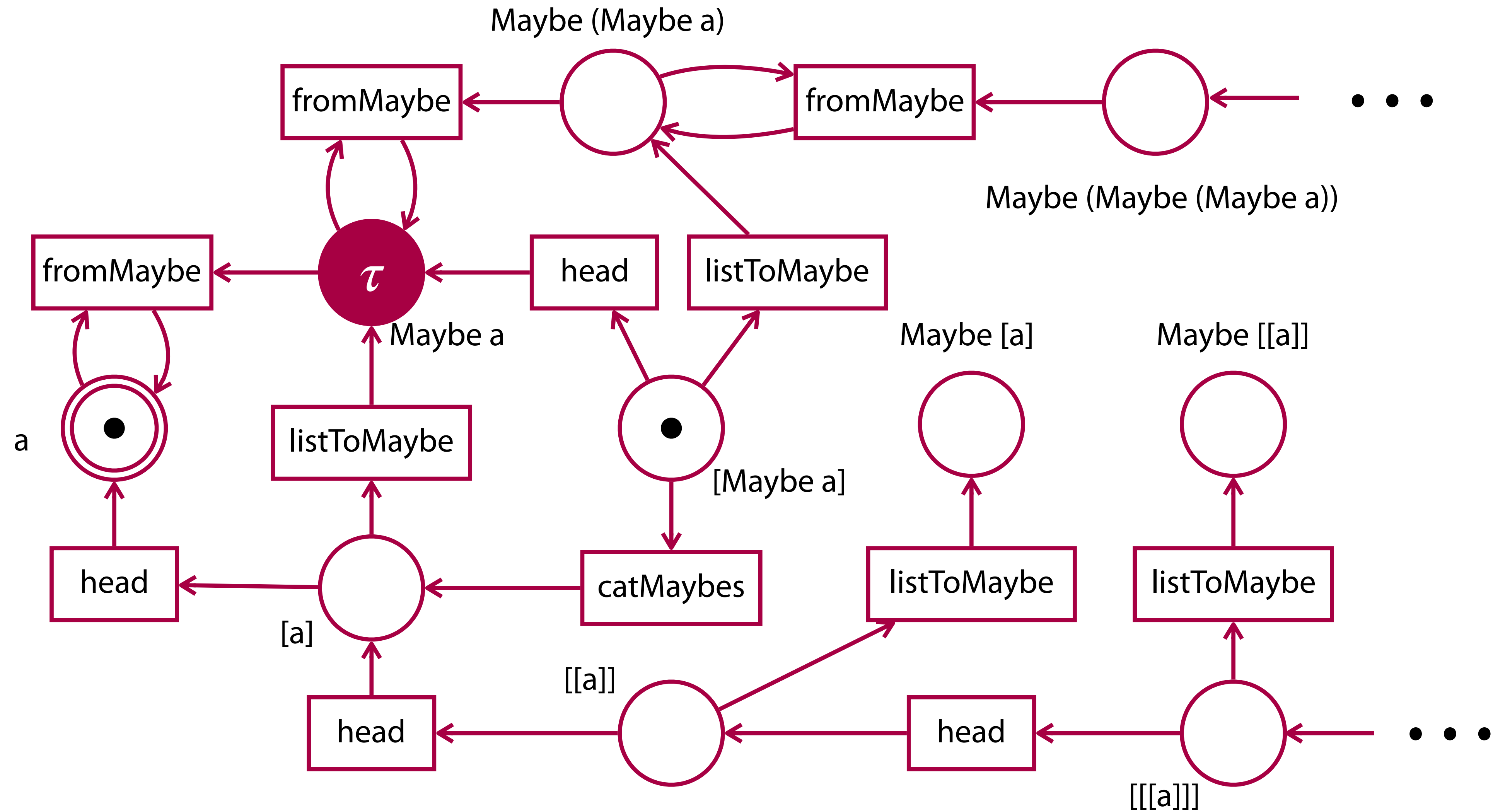
Evaluation

04 Hoogle+: More Features

Type-Guided Abstraction Refinement
Abstract types



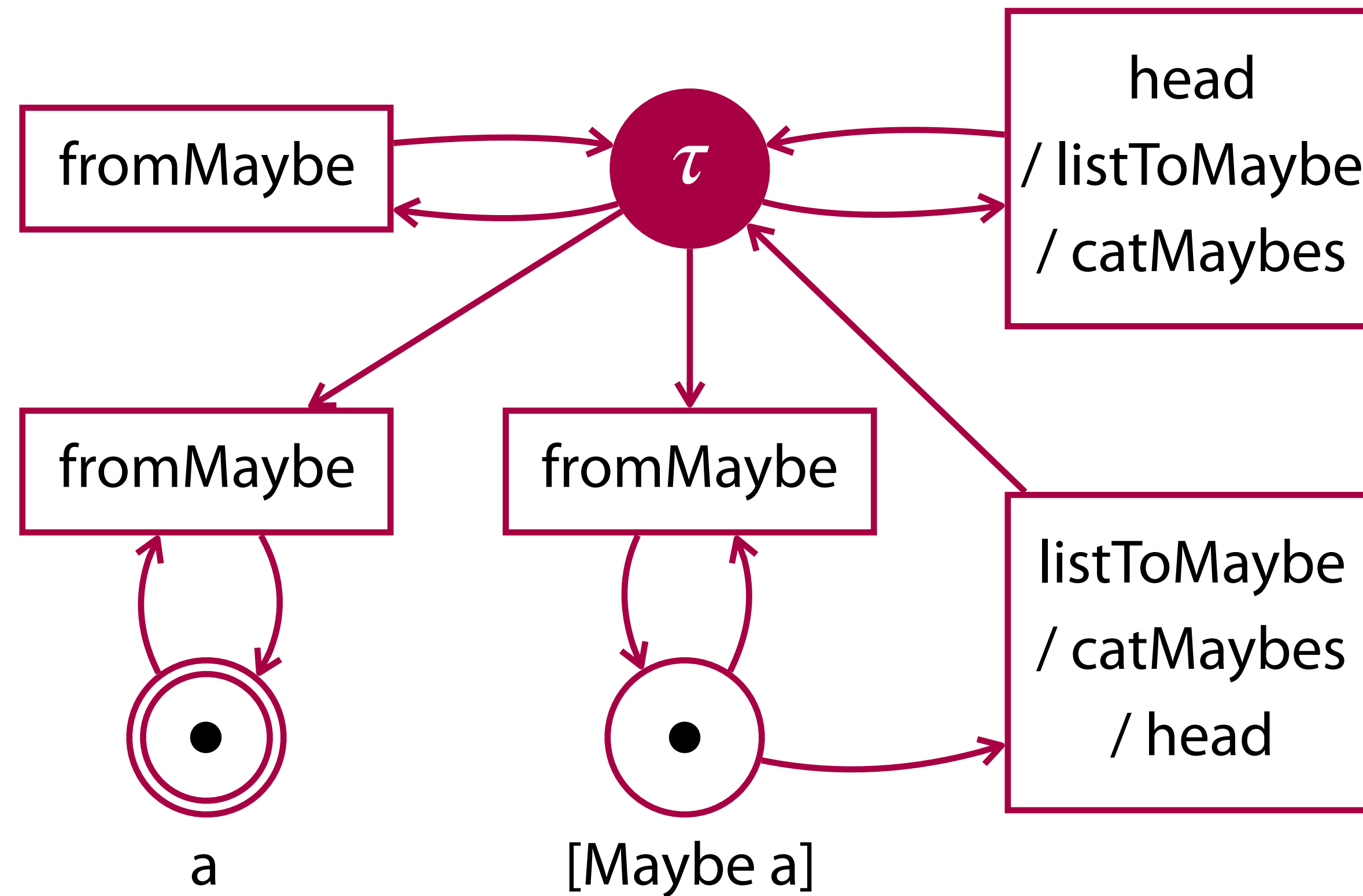
Type-Guided Abstraction Refinement
Abstract petri net



Type-Guided Abstraction Refinement
Abstract petri net

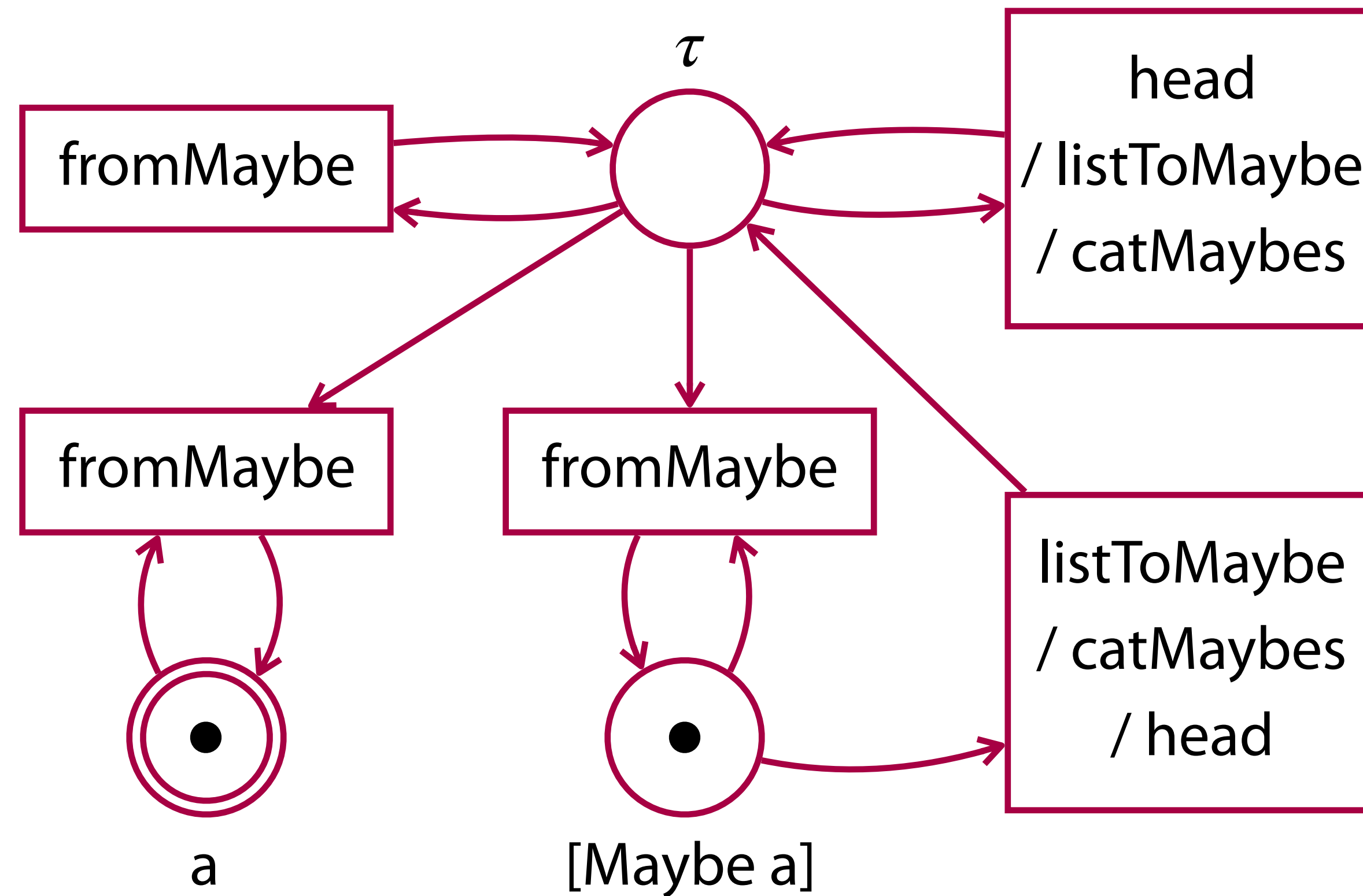
Type Query

d: a -> xs: [Maybe a] -> a



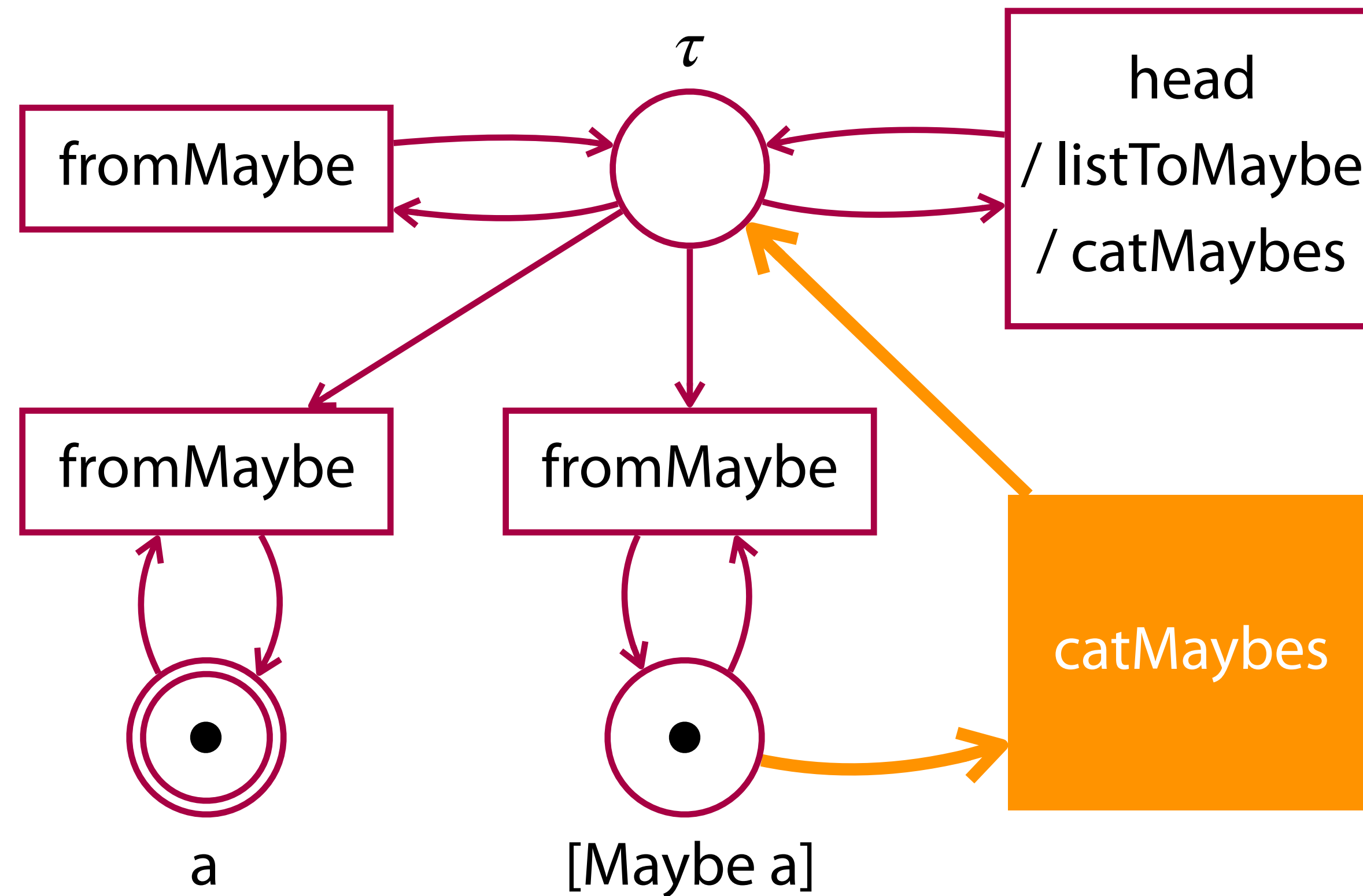
Type-Guided Abstraction Refinement
Abstract petri net

SOLUTION: $\lambda d xs \rightarrow \text{fromMaybe } d (\text{listToMaybe } (\text{catMaybes } xs))$



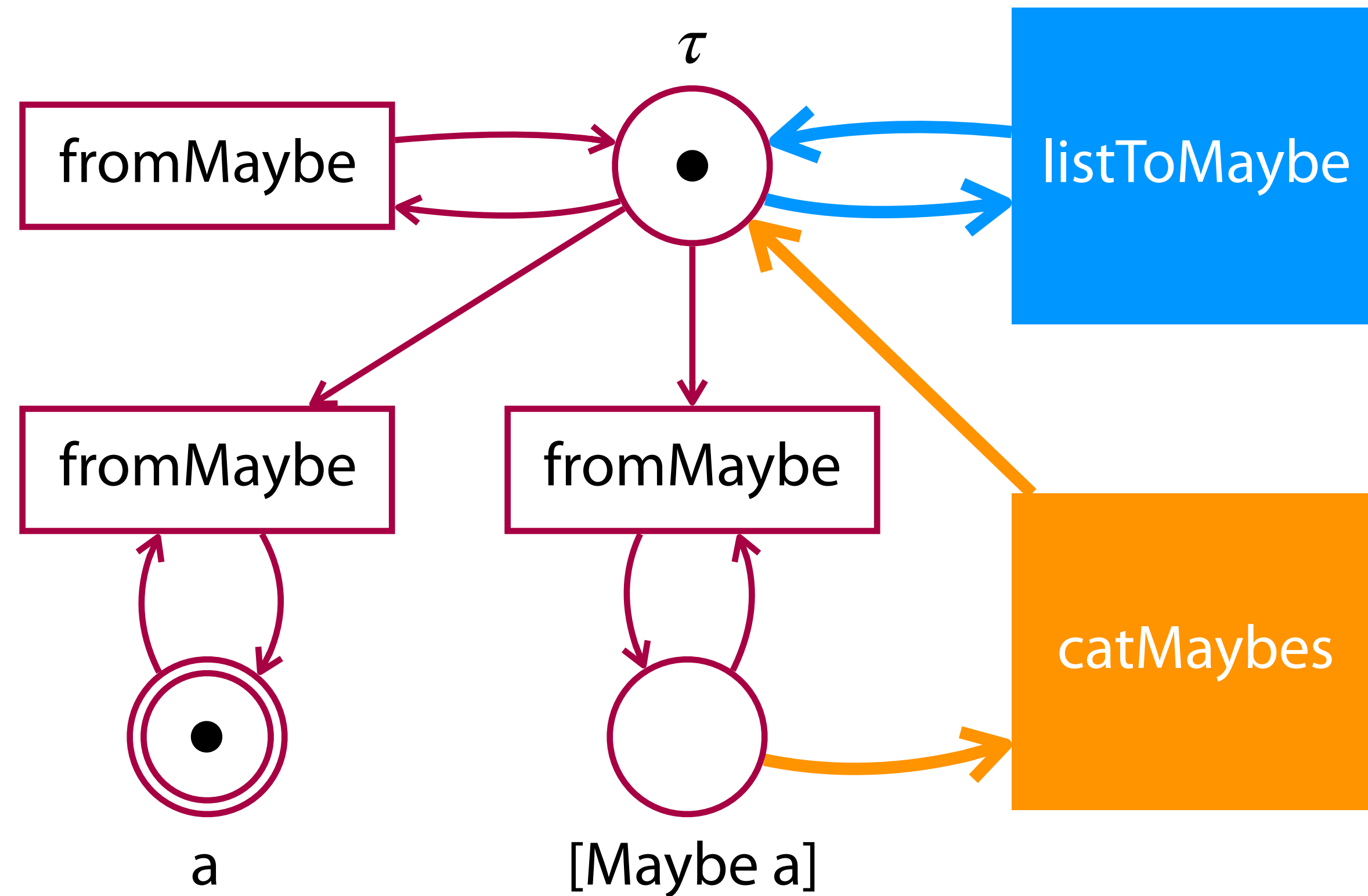
Type-Guided Abstraction Refinement
Abstract petri net

SOLUTION: $\lambda d xs \rightarrow \text{fromMaybe } d (\text{listToMaybe } (\text{catMaybes } xs))$



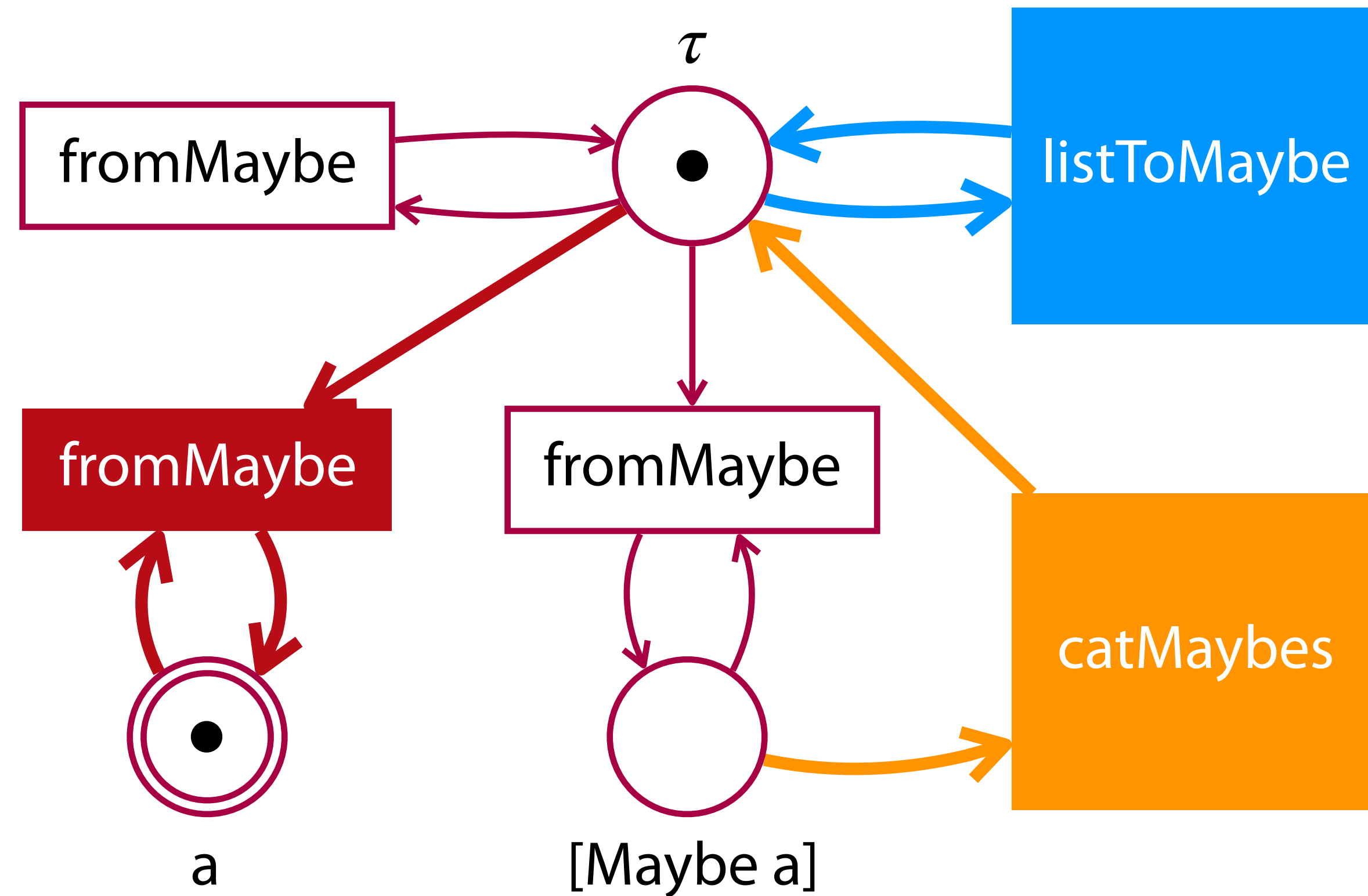
Type-Guided Abstraction Refinement
Abstract petri net

SOLUTION: $\backslash d xs \rightarrow$ **fromMaybe** d (**listToMaybe** (**catMaybes** xs))



Type-Guided Abstraction Refinement
Abstract petri net

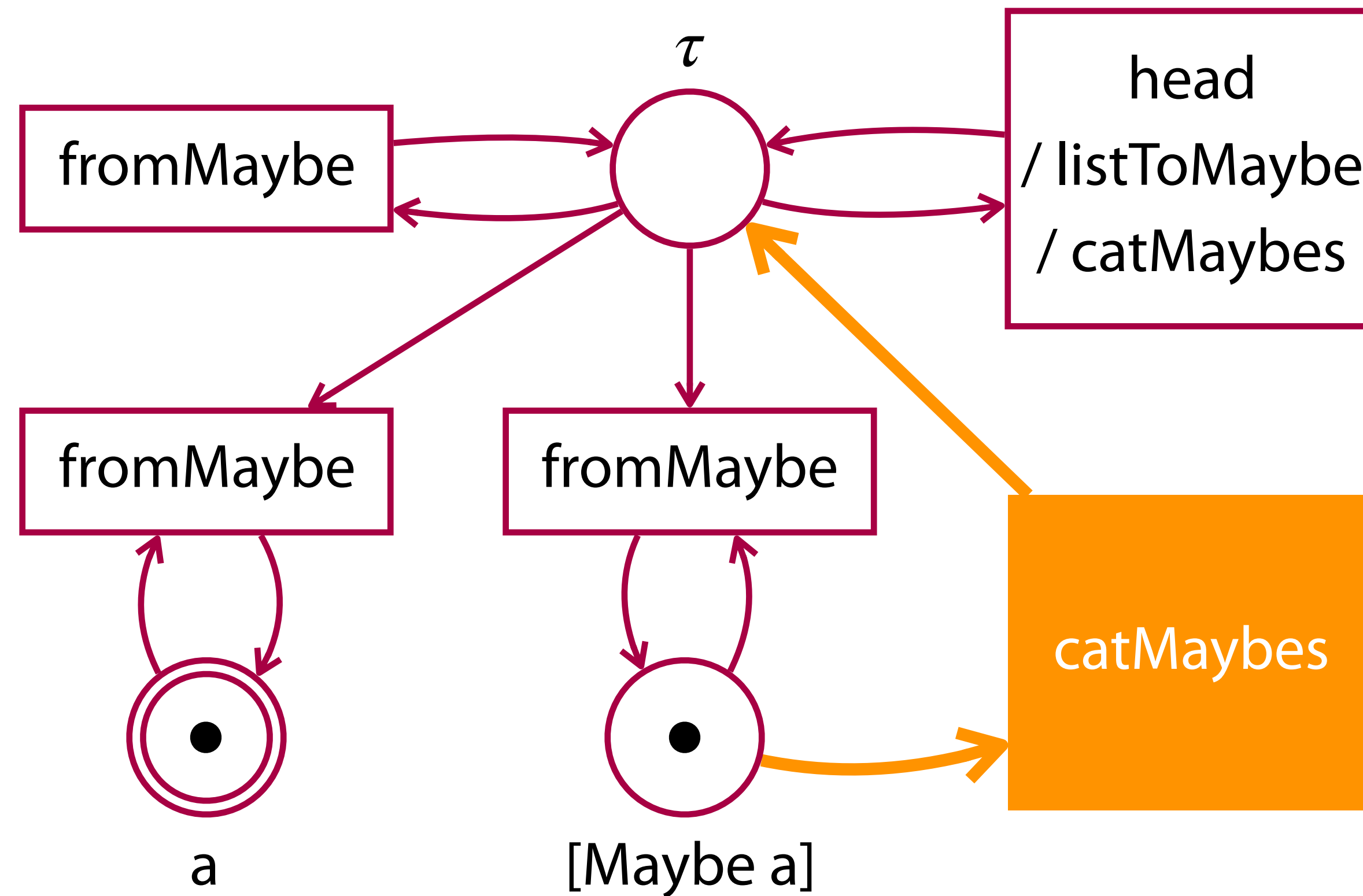
SOLUTION: $\lambda d xs \rightarrow \text{fromMaybe } d (\text{listToMaybe } (\text{catMaybes } xs))$



Type-Guided Abstraction Refinement
Abstract petri net

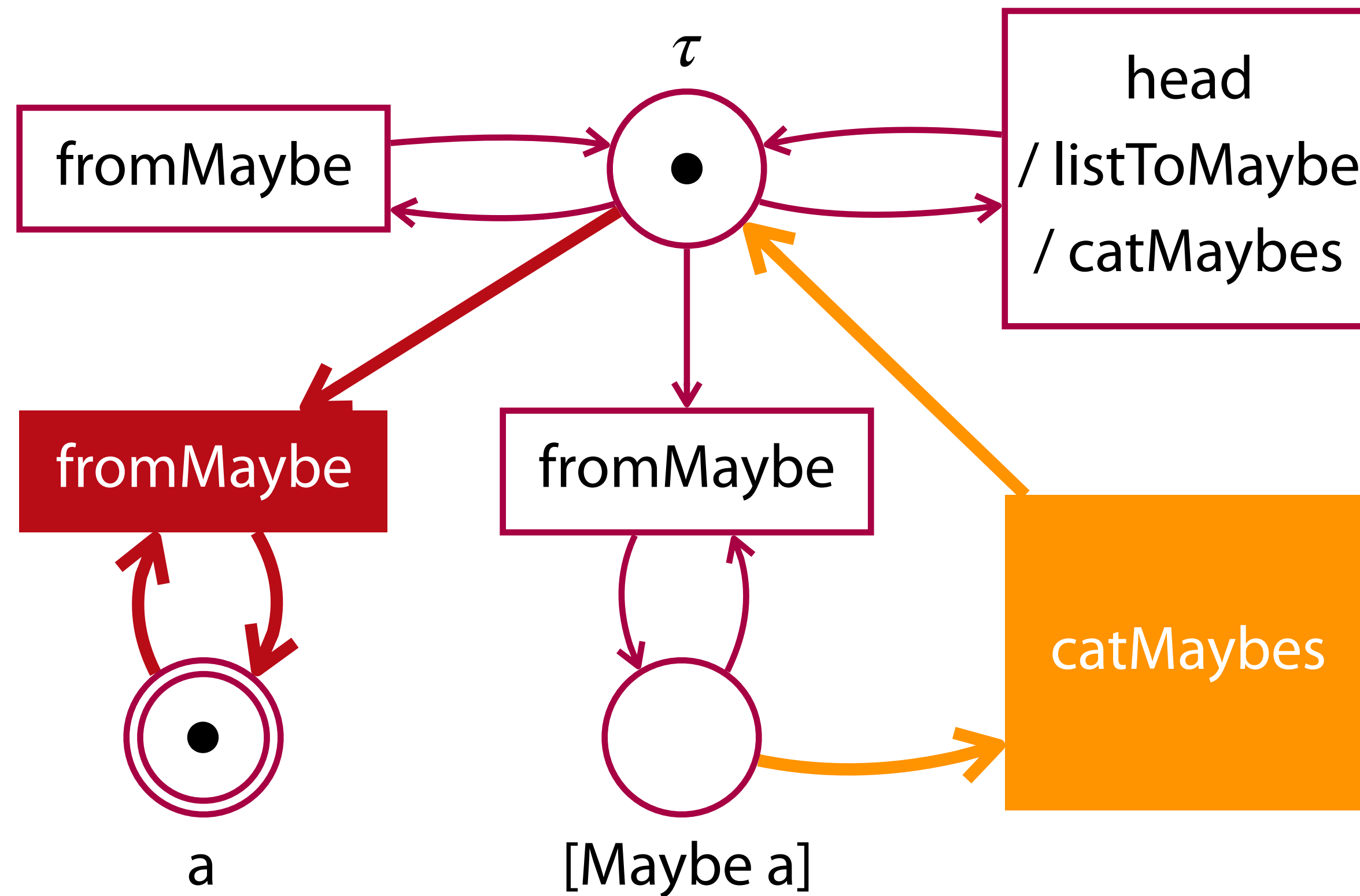
Type Query

$d: a \rightarrow xs: [Maybe a] \rightarrow a$



Type-Guided Abstraction Refinement
Abstract petri net

SOLUTION: $\lambda d xs \rightarrow$ **fromMaybe** d (**catMaybes** xs)

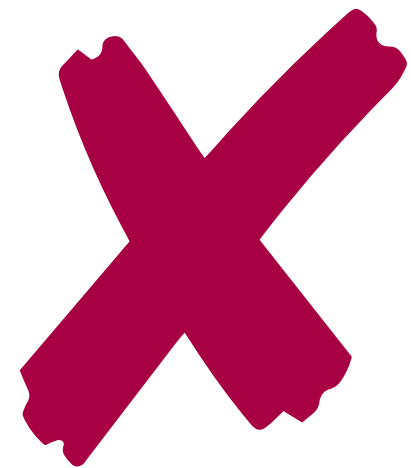


Type-Guided Abstraction Refinement
Abstract petri net

d: Int -> xs: [Maybe Int] -> Int

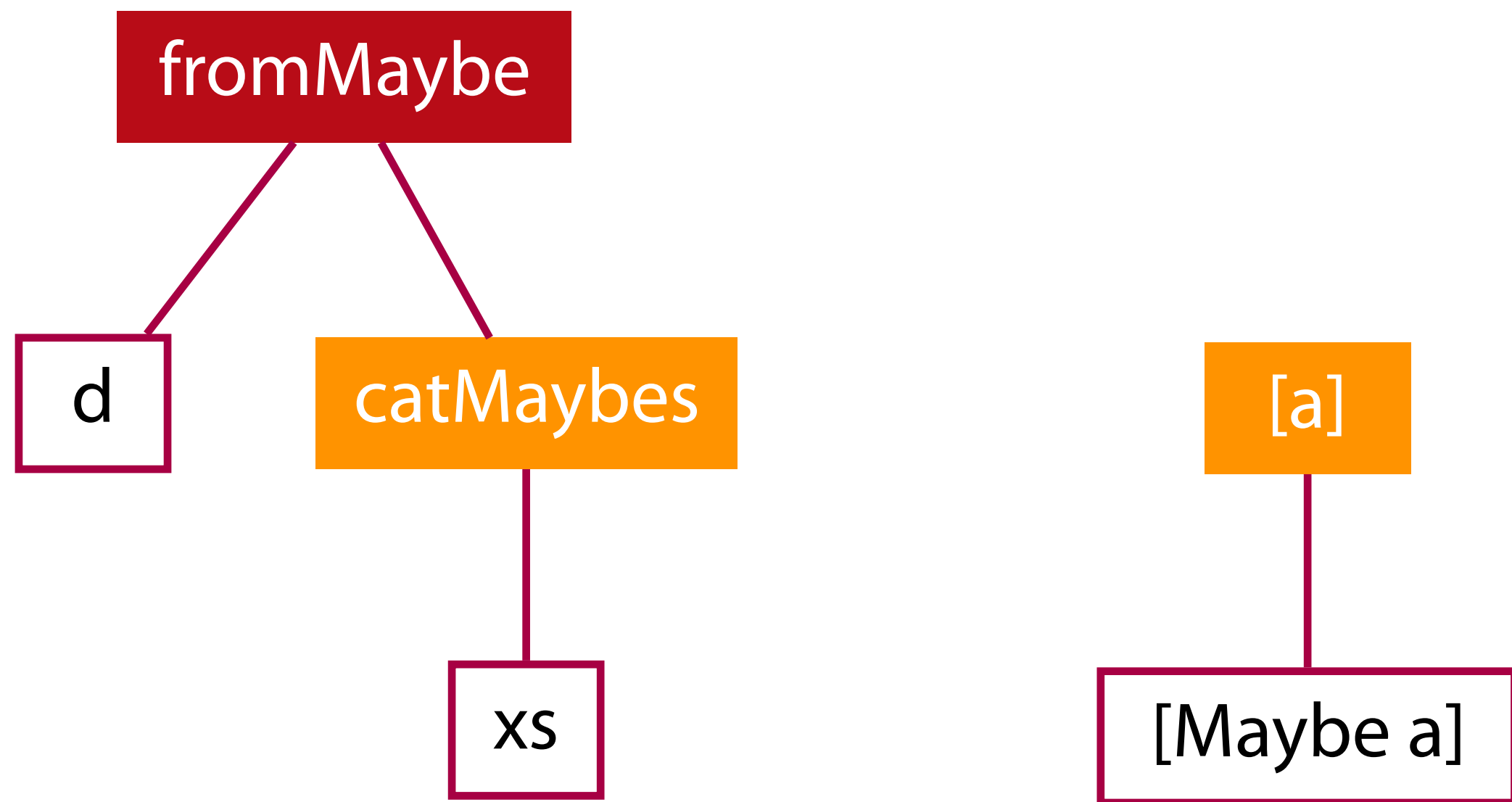


Solution: \d xs -> **fromMaybe** d (**catMaybes** xs)



Type-Guided Abstraction Refinement
Abstract petri net

Spurious Program: \d xs -> **fromMaybe** d (**catMaybes** xs)



AST of the program

Type checking of the program

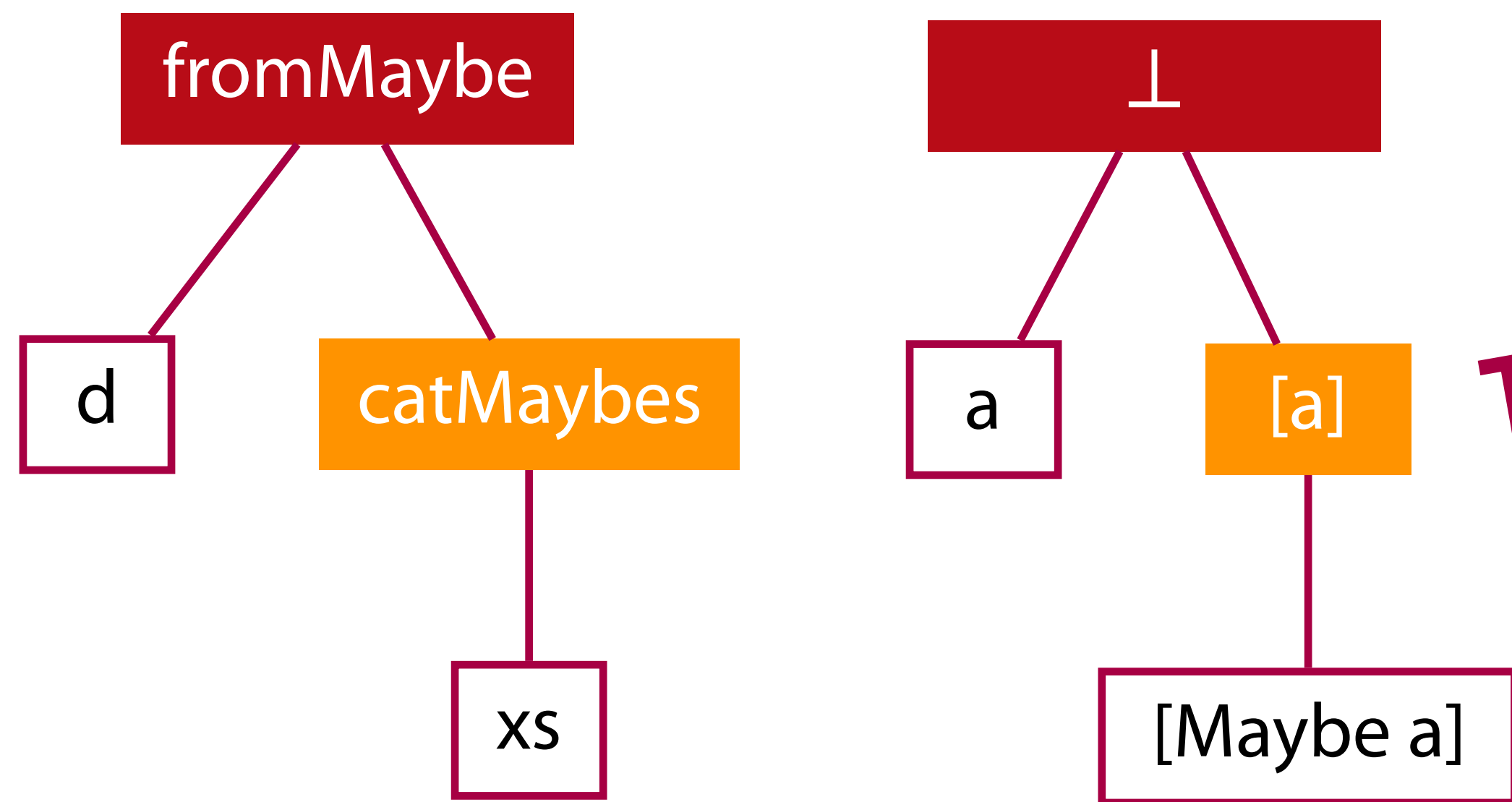
$$\frac{\text{catMaybes} :: \forall \alpha. [\text{Maybe } \alpha] \rightarrow [\alpha] \quad \text{xs} :: [\text{Maybe } a]}{\text{catMaybes xs} :: [a]}$$

Type-Guided Abstraction Refinement
Abstract petri net

Type Query

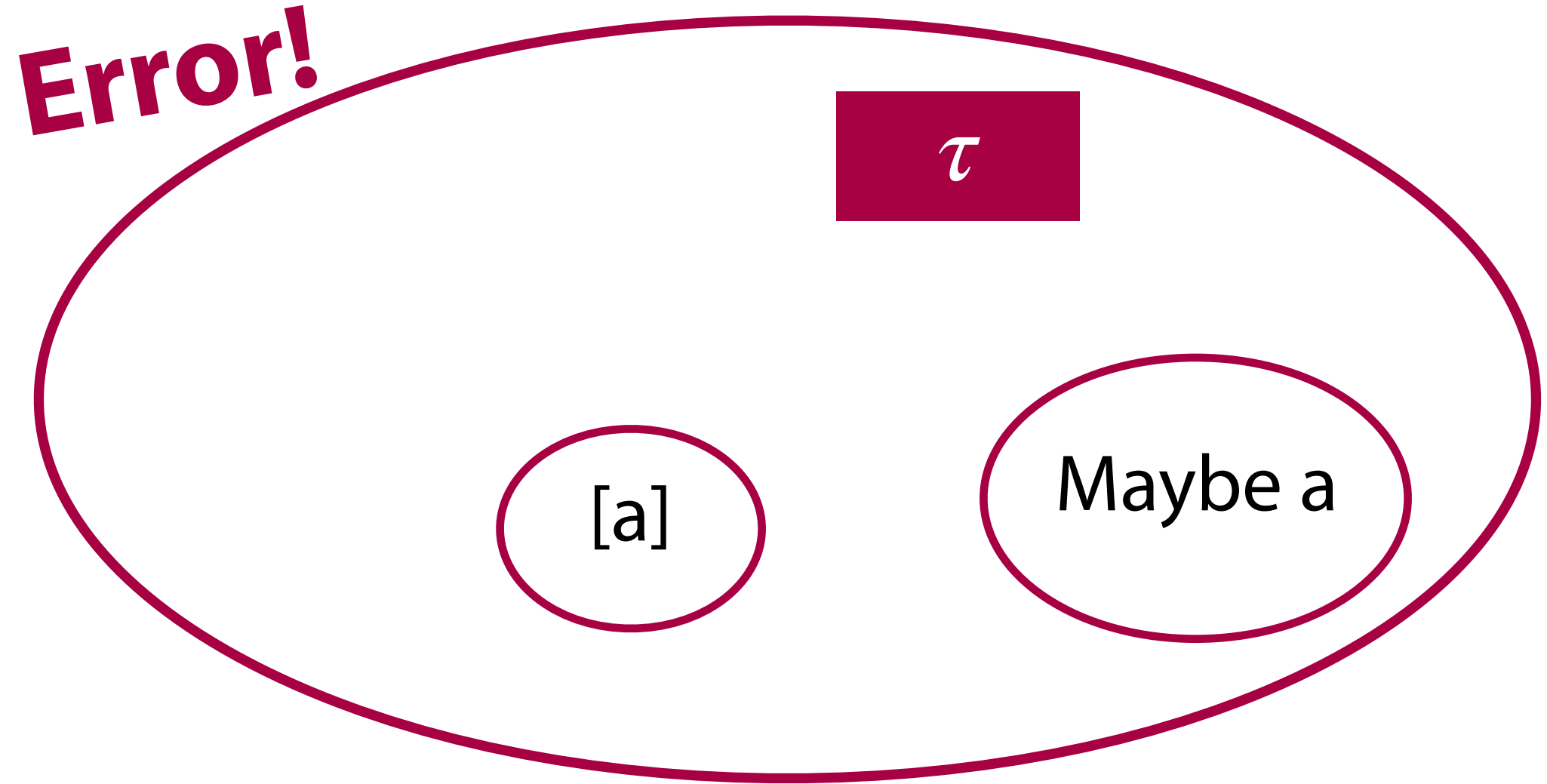
$d: a \rightarrow xs: [Maybe a] \rightarrow a$

Spurious Program: $\lambda d xs \rightarrow \text{fromMaybe } d (\text{catMaybes } xs)$



$\text{fromMaybe} :: \forall \alpha. \alpha \rightarrow \text{Maybe } \alpha \rightarrow \alpha$

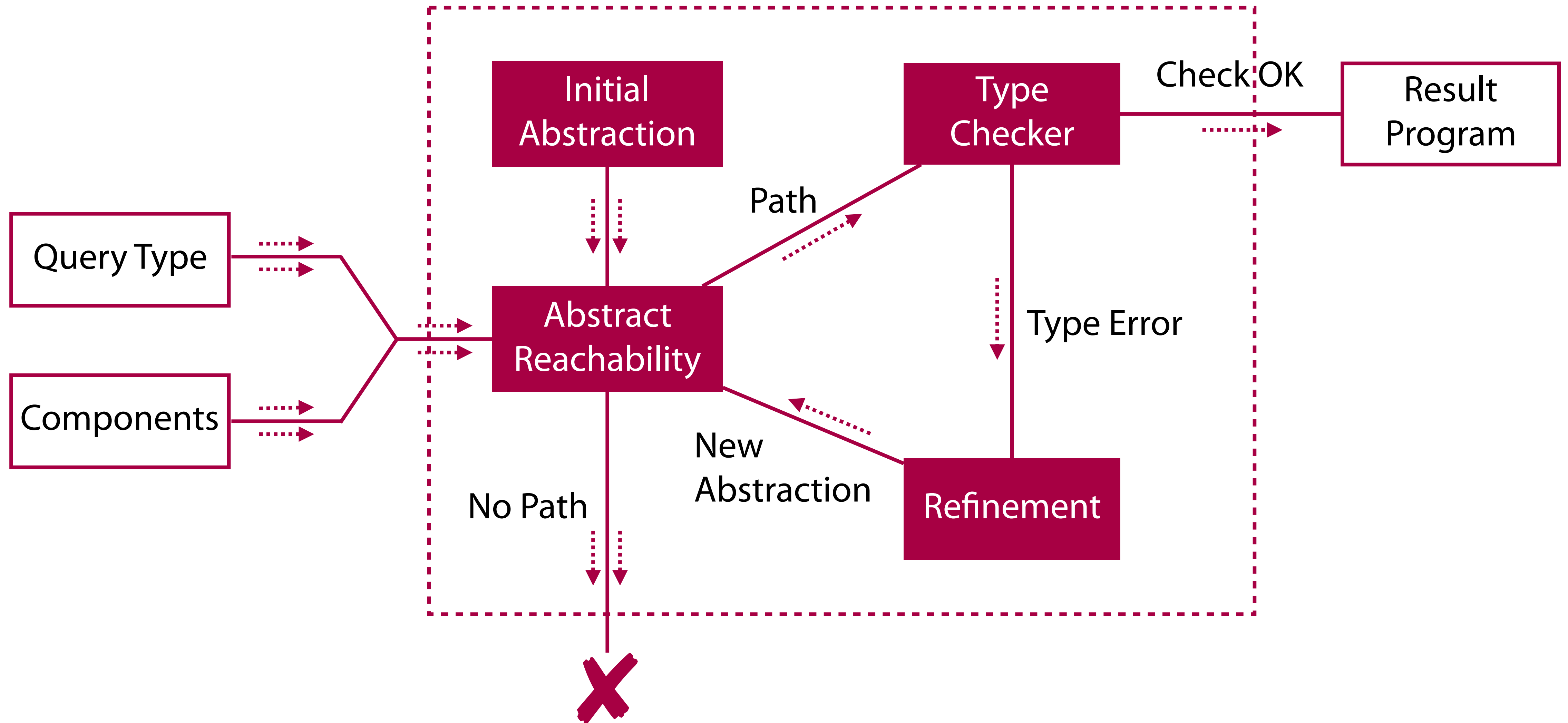
Type Error!



AST of the program

Type checking of the program

Type-Guided Abstraction Refinement TyGAR Workflow



Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: **Type-Guided Abstraction Refinement**

Abstraction

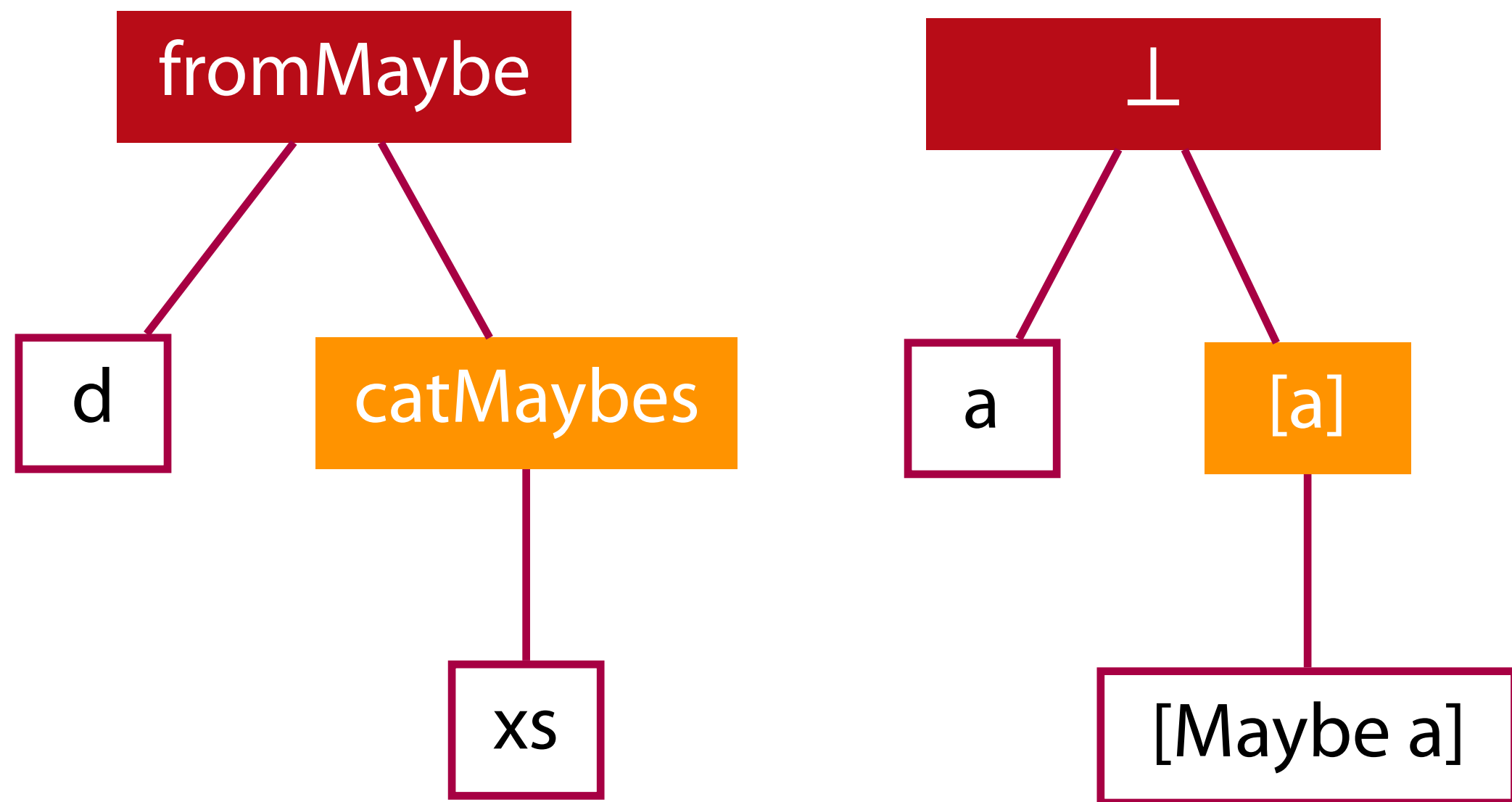
Refinement

Evaluation

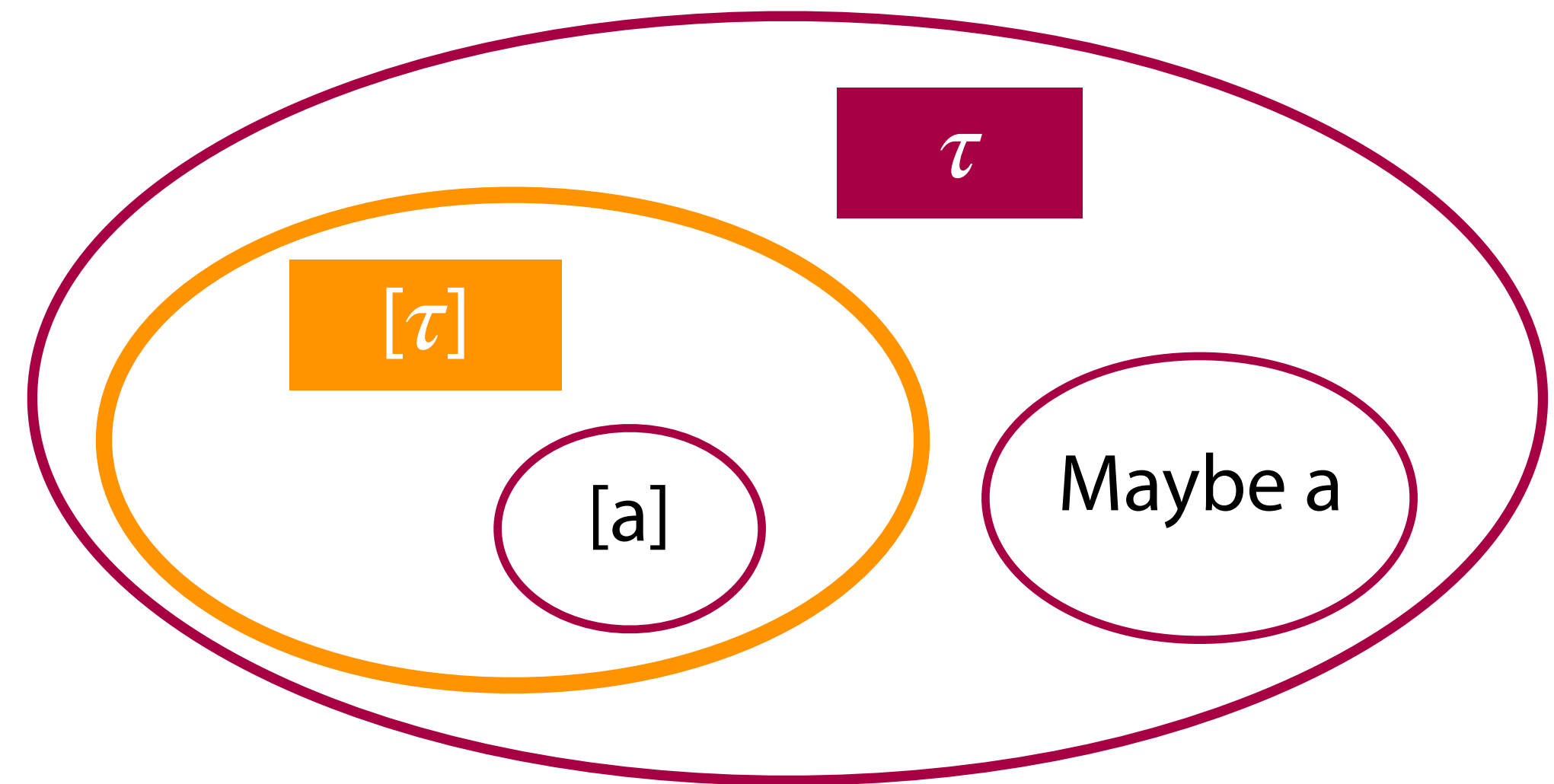
04 Hoogle+: More Features

Type-Guided Abstraction Refinement Type abstraction refinement

Spurious Program: \d xs -> **fromMaybe** d (**catMaybes** xs)



fromMaybe :: $\forall \alpha. \alpha \rightarrow$ Maybe α $\rightarrow \alpha$



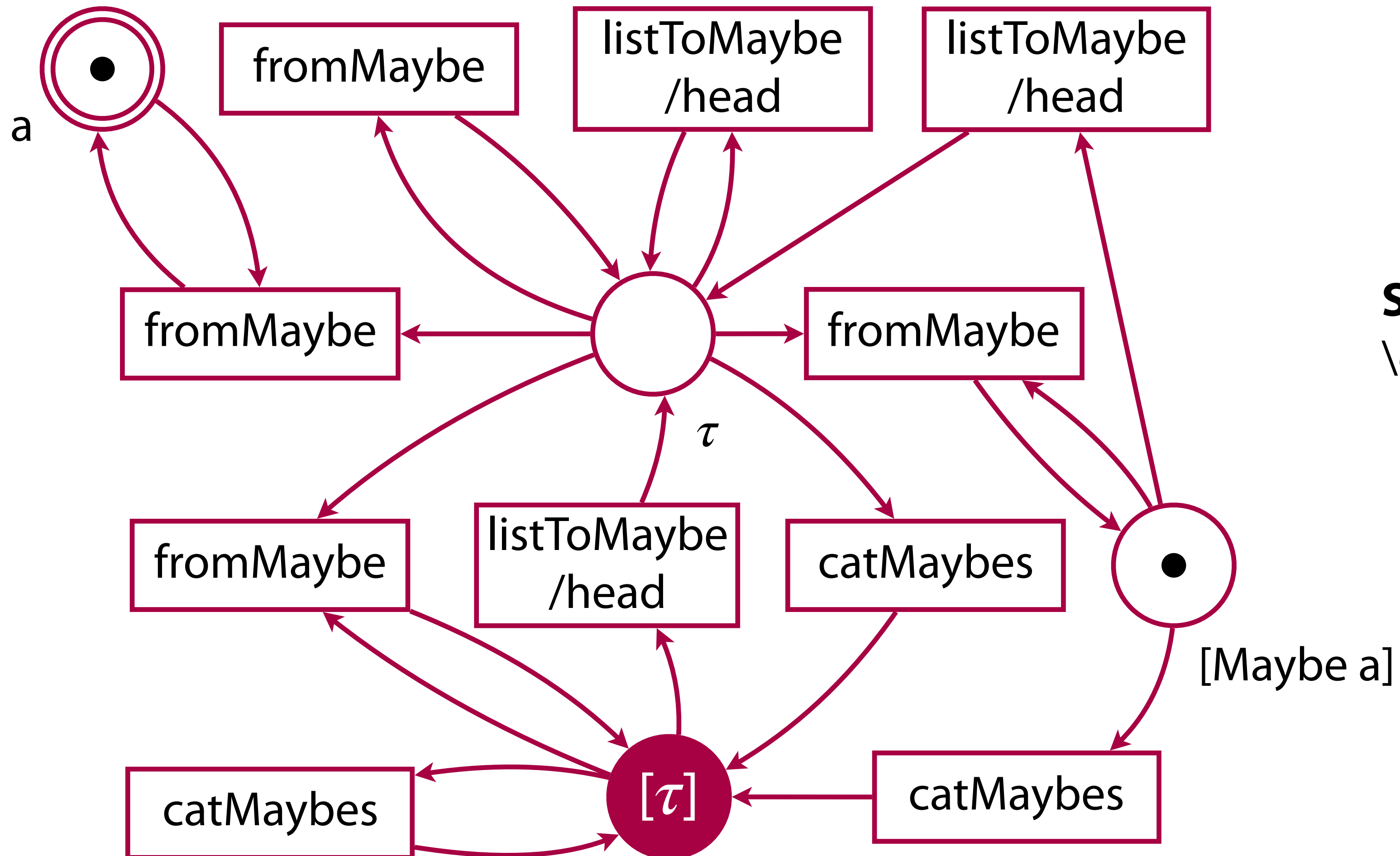
AST of the program

Type checking of the program

Type Query

$d: a \rightarrow xs: [Maybe a] \rightarrow a$

Type-Guided Abstraction Refinement
Refined abstract petri net



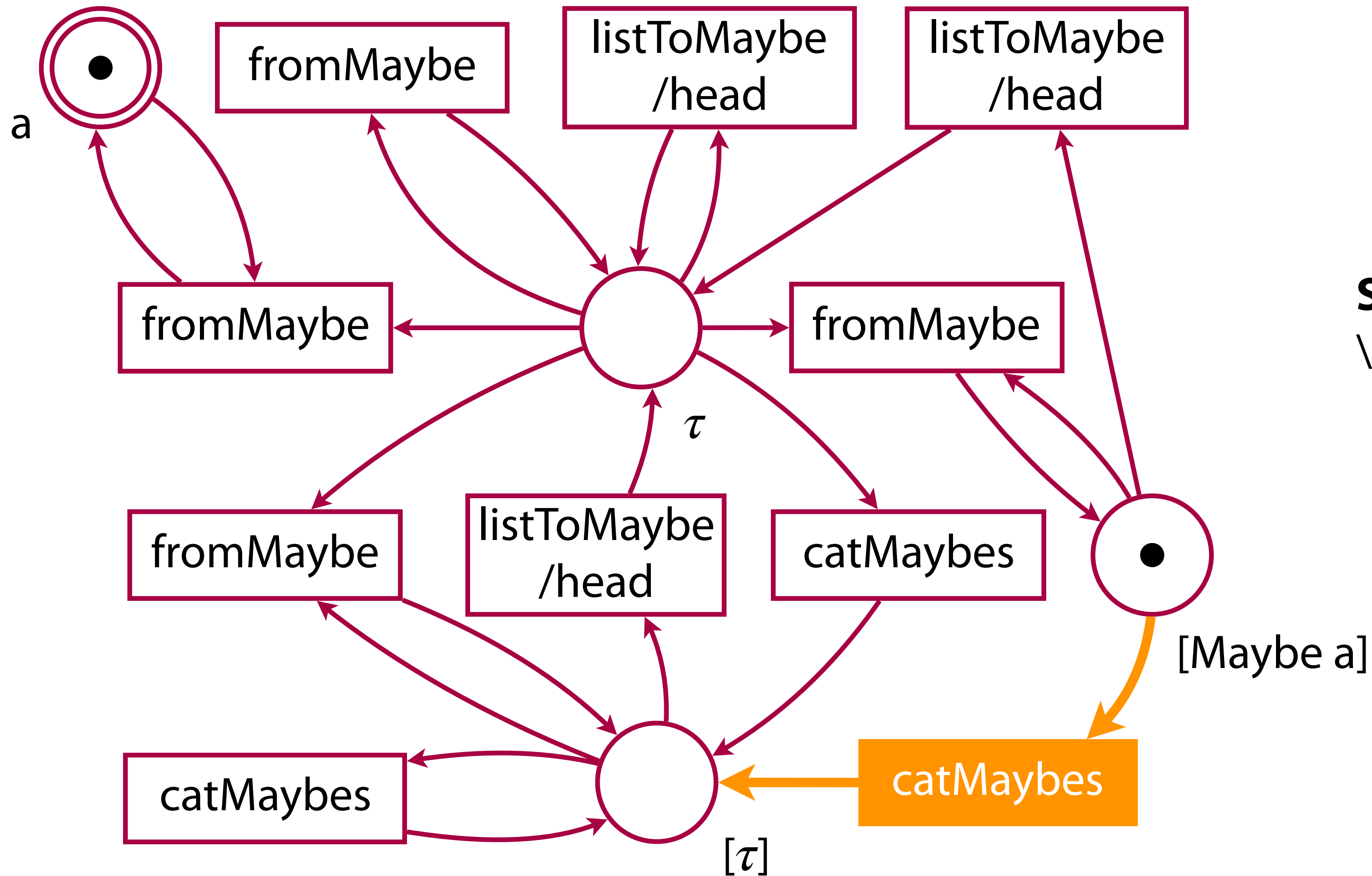
Spurious Program:

$\backslash d xs \rightarrow$ **fromMaybe** d (**catMaybes** xs)

Type Query

$d: a \rightarrow xs: [Maybe a] \rightarrow a$

Type-Guided Abstraction Refinement
Refined abstract petri net



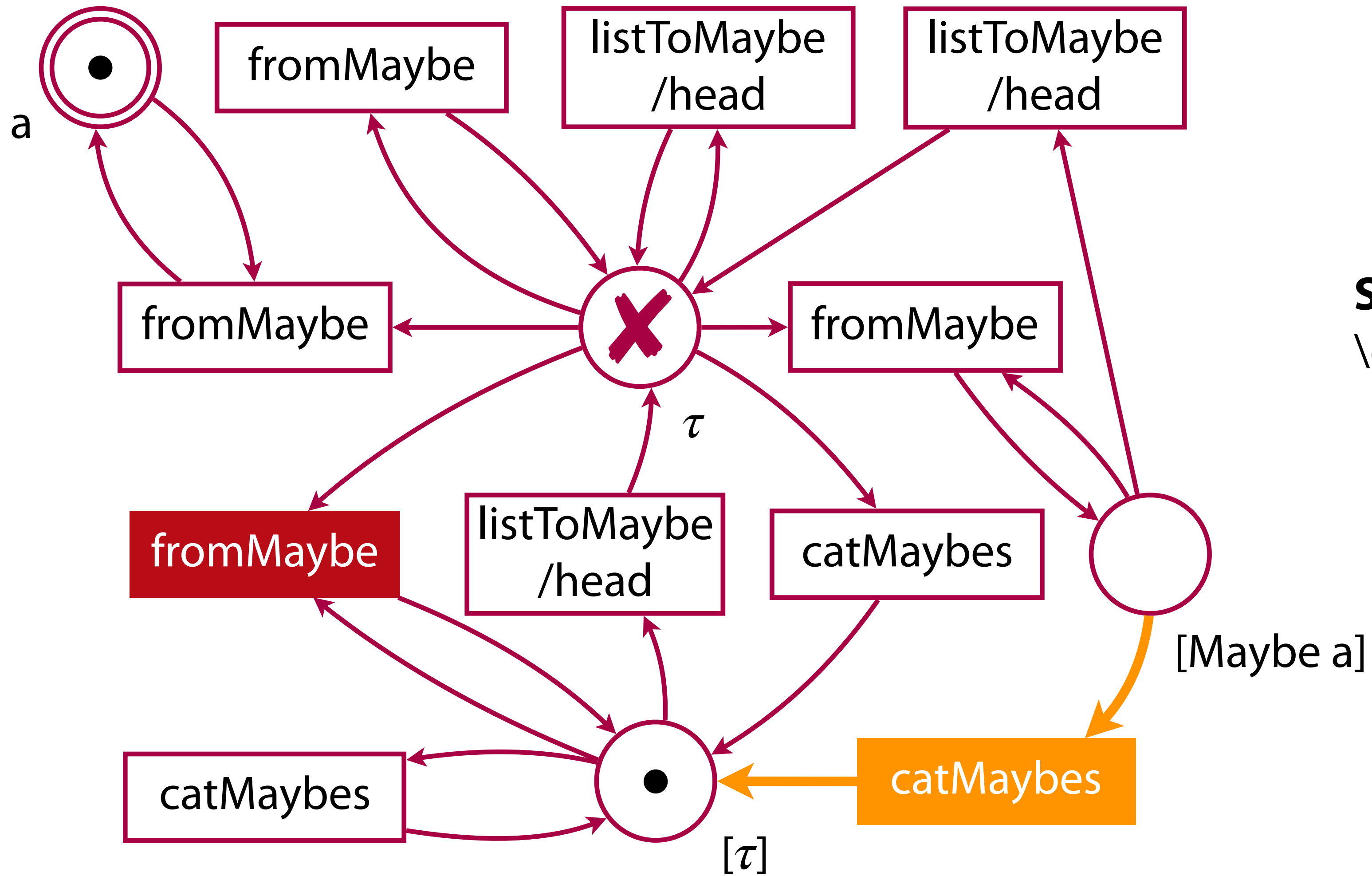
Spurious Program:

$\backslash d \ xs \rightarrow \text{fromMaybe } d \ (\text{catMaybes } xs)$

Type Query

$d: a \rightarrow xs: [Maybe a] \rightarrow a$

Type-Guided Abstraction Refinement
Refined abstract petri net



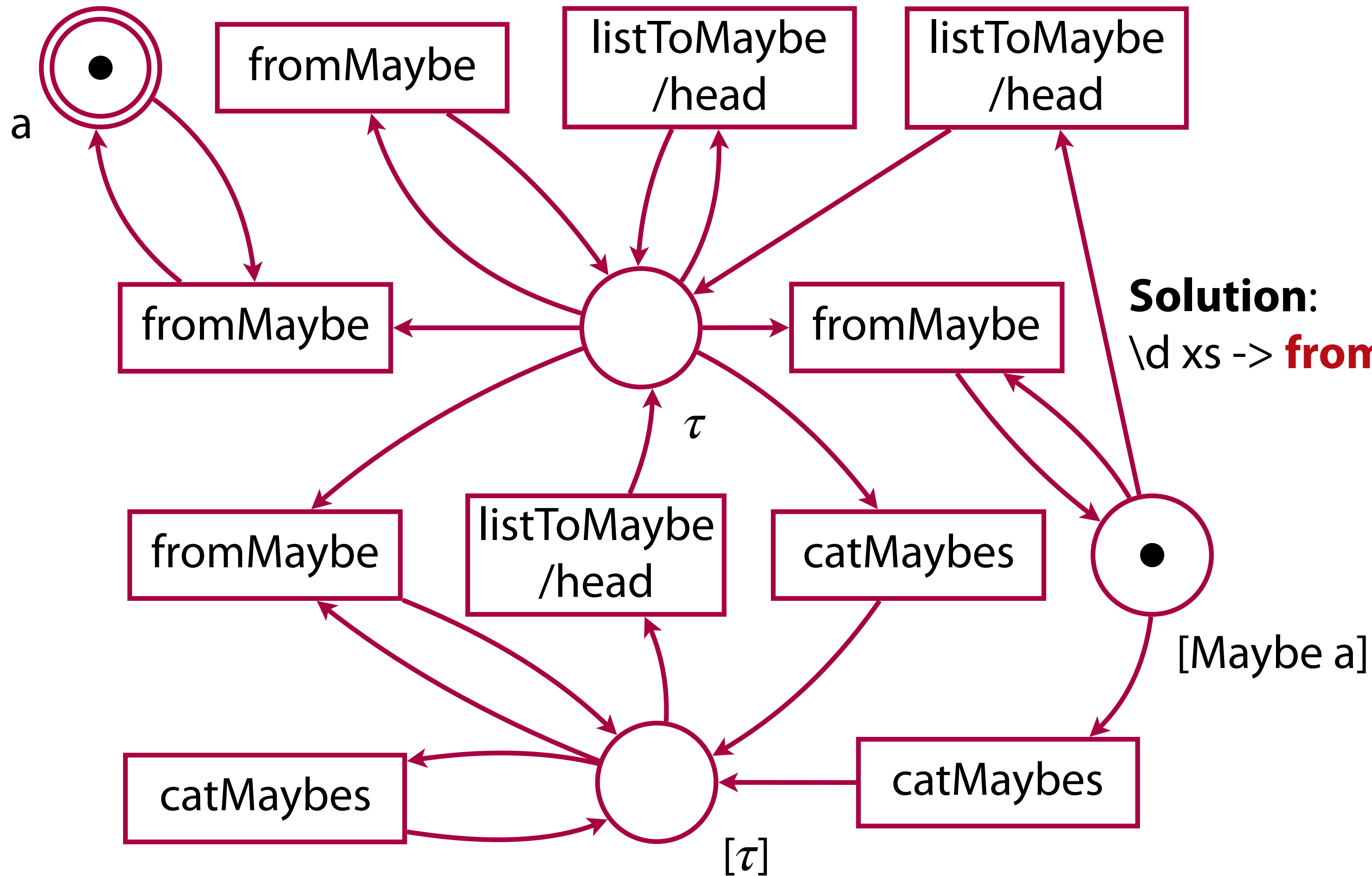
Spurious Program:

$\backslash d xs \rightarrow$ **fromMaybe** d (**catMaybes** xs)

Type Query

$d: a \rightarrow xs: [Maybe a] \rightarrow a$

Type-Guided Abstraction Refinement
Refined abstract petri net



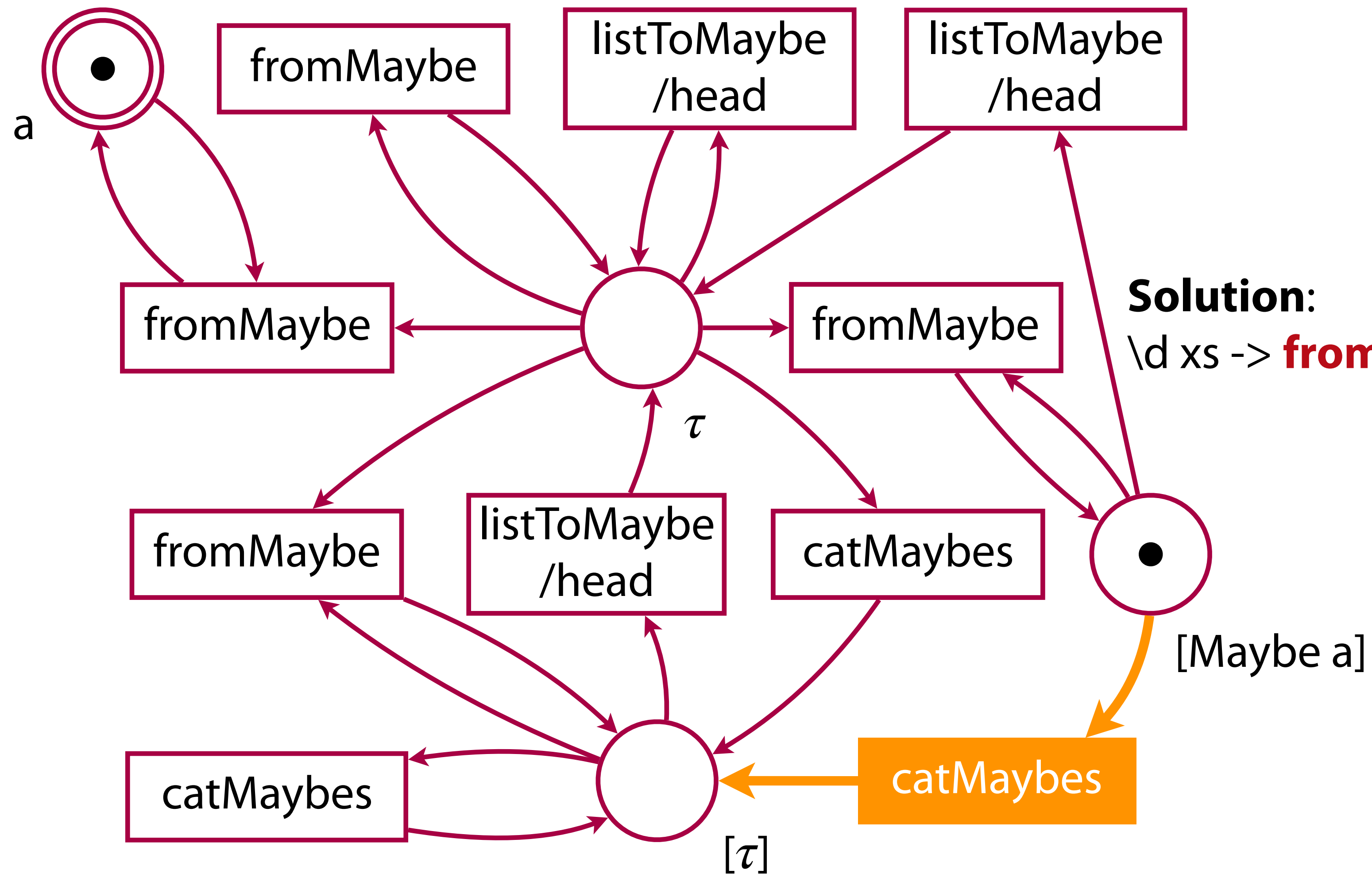
Solution:

$\backslash d xs \rightarrow$ **fromMaybe** d (**listToMaybe** (**catMaybes** xs))

Type Query

$d: a \rightarrow xs: [Maybe a] \rightarrow a$

Type-Guided Abstraction Refinement
Refined abstract petri net



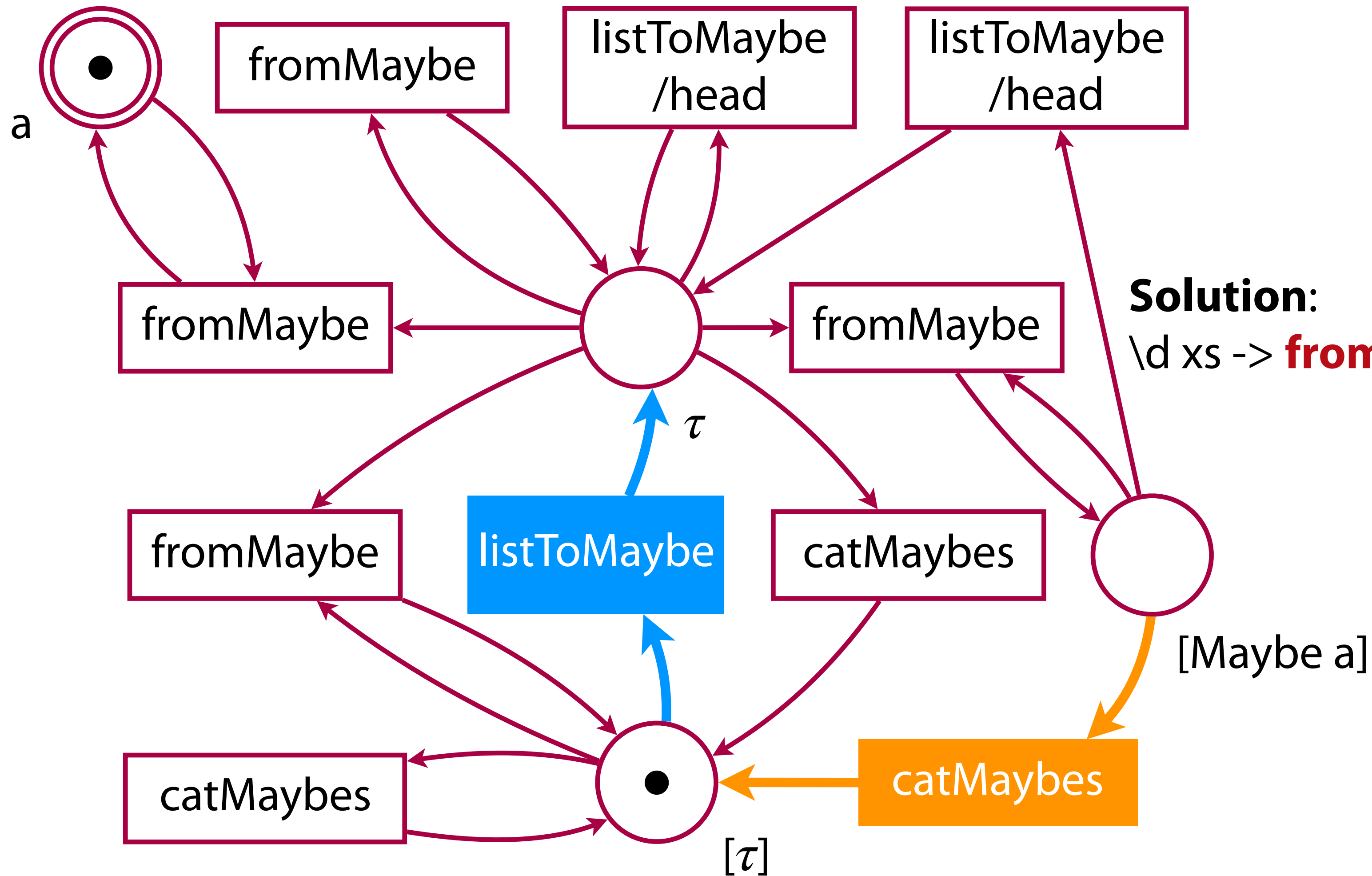
Solution:

$\backslash d xs \rightarrow$ **fromMaybe** d (**listToMaybe** (**catMaybes** xs))

Type Query

$d: a \rightarrow xs: [Maybe a] \rightarrow a$

Type-Guided Abstraction Refinement
Refined abstract petri net



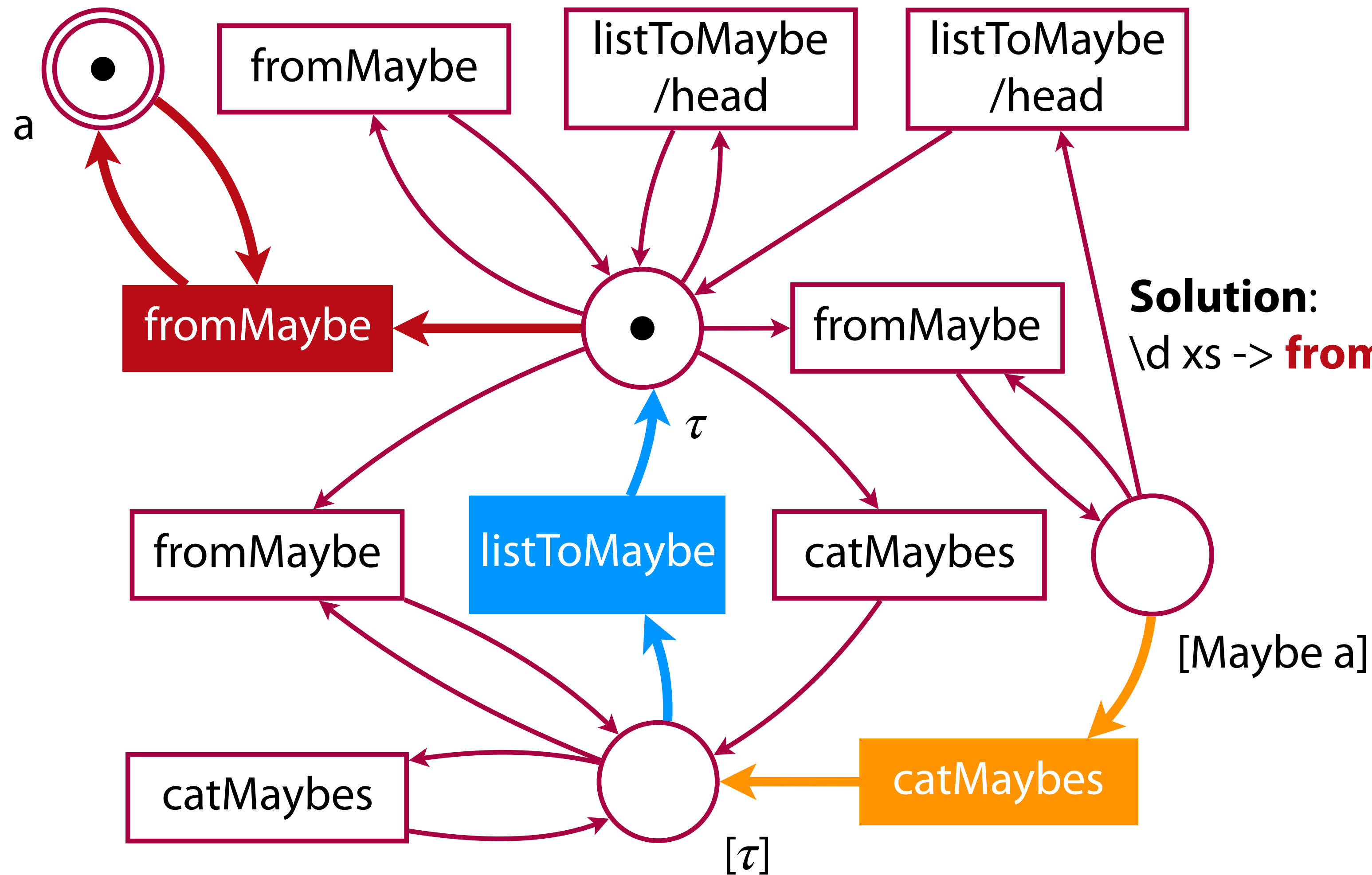
Solution:

$\backslash d xs \rightarrow$ **fromMaybe** d (**listToMaybe** (**catMaybes** xs))

Type Query

$d: a \rightarrow xs: [Maybe a] \rightarrow a$

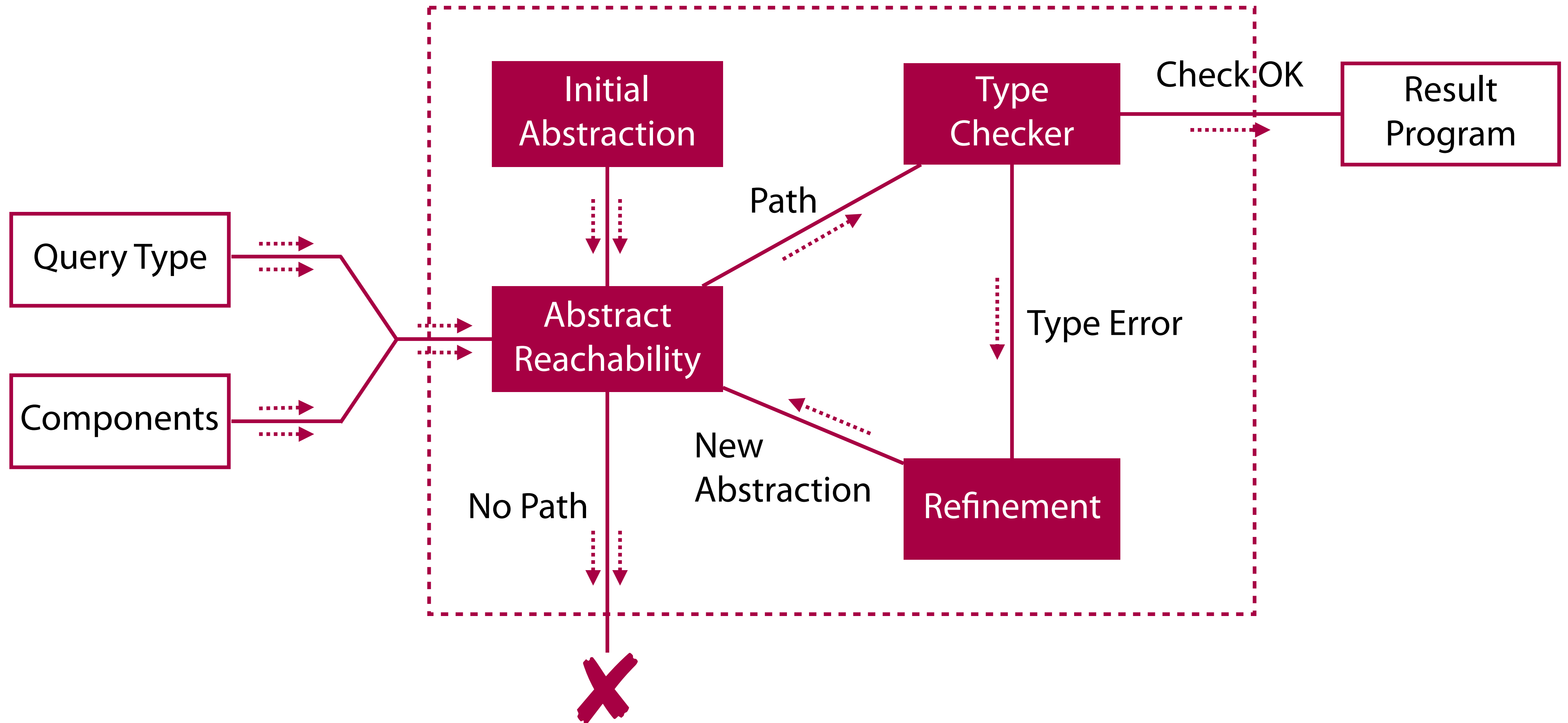
Type-Guided Abstraction Refinement
Refined abstract petri net



Solution:

$\backslash d xs \rightarrow$ **fromMaybe** d (**listToMaybe** (**catMaybes** xs))

Type-Guided Abstraction Refinement TyGAR Workflow



Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: **Type-Guided Abstraction Refinement**

Abstraction

Refinement

Evaluation

04 Hoogle+: More Features

Type-Guided Abstraction Refinement Evaluation

Components

291 components

12 popular Haskell library modules

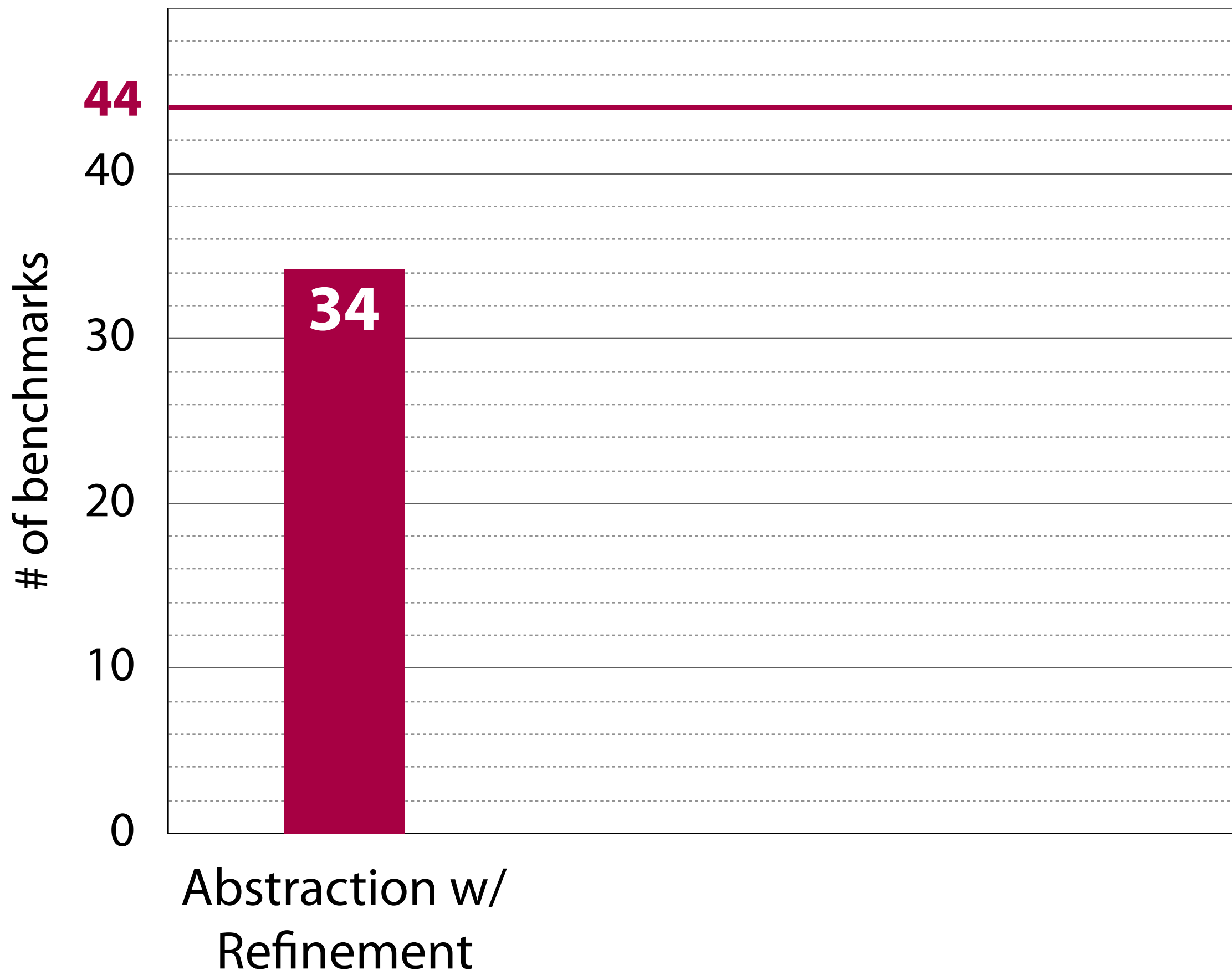
Benchmarks

24 benchmarks from Hoogle

6 benchmarks from StackOverflow

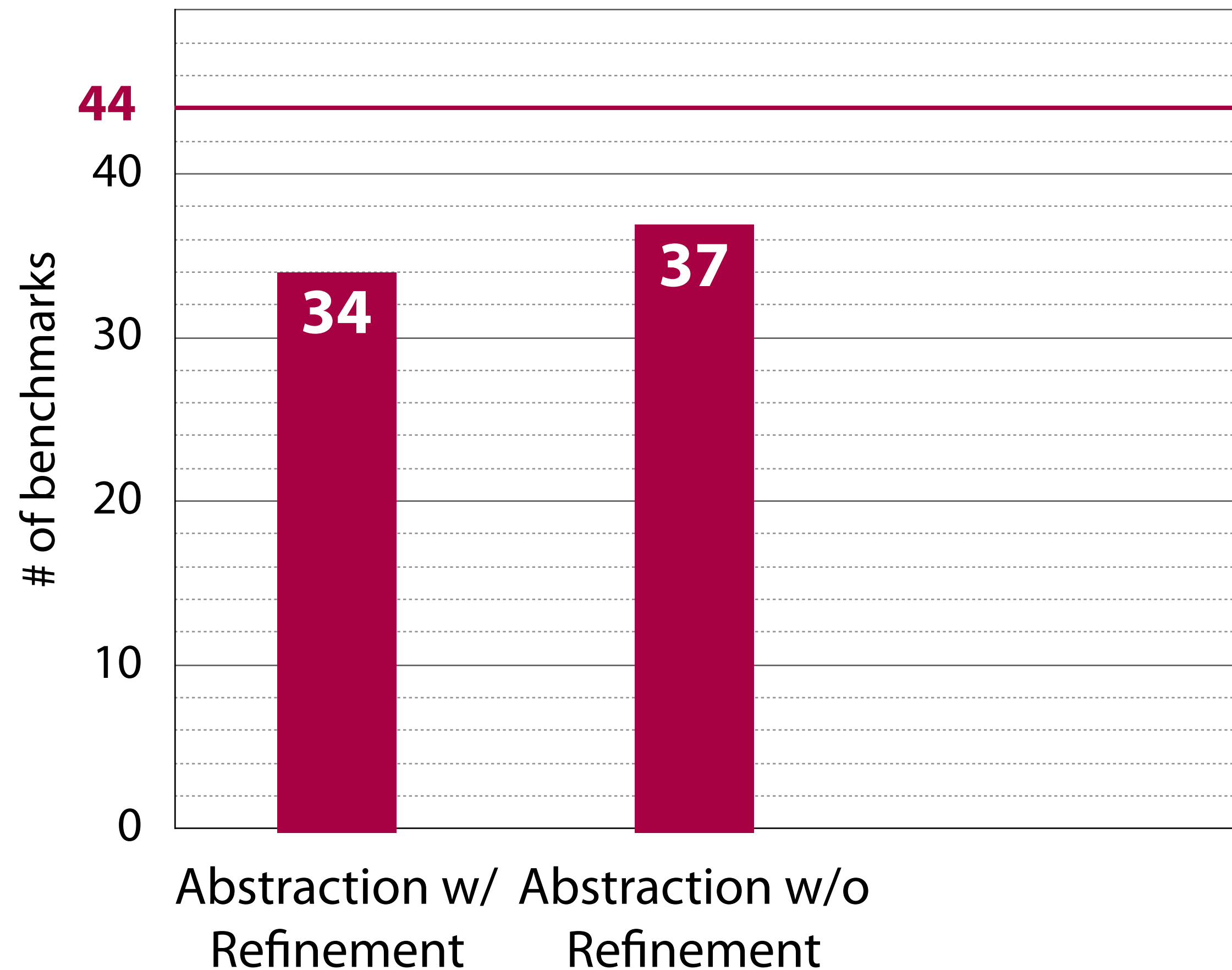
14 benchmarks curated by us

Type-Guided Abstraction Refinement Evaluation



Too many refinements → Too large petri net

Type-Guided Abstraction Refinement Evaluation



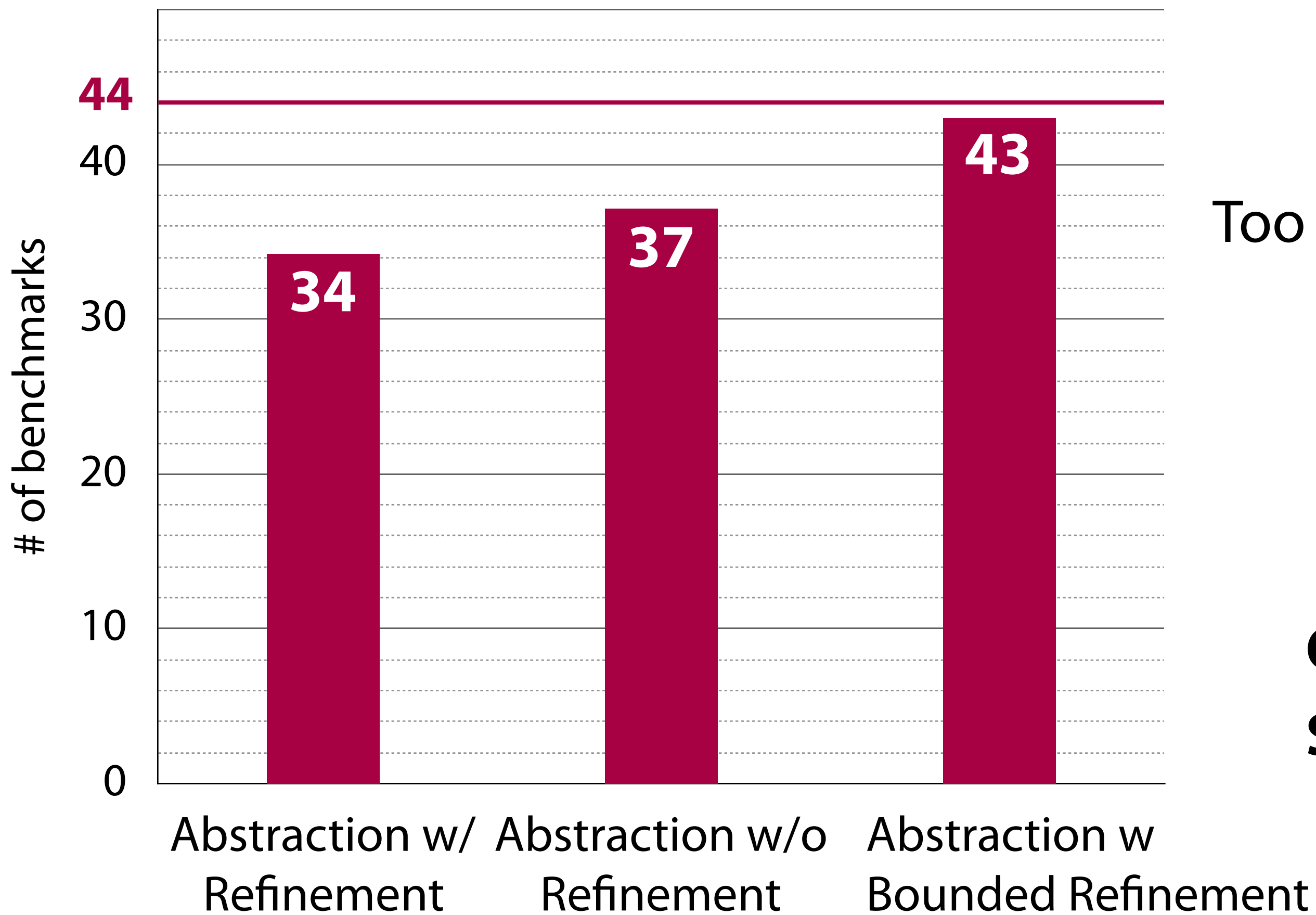
Too many refinements → Too large petri net
No refinement → Poor at hard queries

Query: $f: (a \rightarrow b) \rightarrow g: (a \rightarrow c) \rightarrow x: a \rightarrow (b, c)$

Solution: $\backslash f g x \rightarrow (f x, g x)$

TIMEOUT!

Type-Guided Abstraction Refinement Evaluation



Too many refinements → Too large petri net
No refinement → Poor at hard queries

Query: $f: (a \rightarrow b) \rightarrow g: (a \rightarrow c) \rightarrow x: a \rightarrow (b, c)$

Solution: $\backslash f g x \rightarrow (f x, g x)$

Bounded refinements 2s!

Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: Type-Guided Abstraction Refinement

Abstraction

Refinement

Evaluation

04 Hoogle+: More Features



Hoogle+

Support for real-world Haskell

More advanced Haskell features

higher-order functions, type classes, etc.

Filter out uninteresting solutions

e.g. `\d xs -> fromLeft d (Right xs)` always returns **d**

Outline

01 Problem: Component-Based Synthesis

Running example

Previous solution: SyPet

02 Challenge: Polymorphism

Search space explosion

03 Solution: **Type-Guided Abstraction Refinement**

Abstraction

Refinement

Evaluation

04 Hoogle+: More Features