APIphany: Type-Directed Program Synthesis For REST APIs

Zheng Guo*, David Cao*, Davin Tjong*, Jean Yang†, Cole Schlesinger†, Nadia Polikarpova*

* University of California San Diego

† Akita Software

Slack API: Retrieve all member emails from a slack channel

Asked 5 years, 4 months ago Modified 13 days ago Viewed 24k times

16

Given the name of a slack channel, is there a way to retrieve a list of emails of all the members in that channel? I tried looking in the slack api docs but couldn't find the method I need to make this happen (https://api.slack.com/methods).

email channel slack-api slack

1

Share Improve this question Follow

asked Jan 10, 2017 at 8:10

user5844628

377 ● 1 ● 4 ● 12

Task: retrieve all member emails from a Slack channel given the channel name

```
channel_name ⇒ {
  conversations_list()




  }
```

Task: retrieve all member emails from a Slack channel given the channel name

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)




  }
```

Task: retrieve all member emails from a Slack channel given the channel name

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)



  })}
```

Task: retrieve all member emails from a Slack channel given the channel name

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                                          })
  })}
```

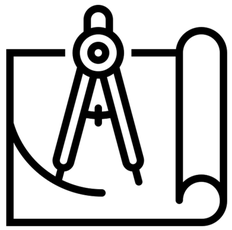
Task: retrieve all member emails from a Slack channel given the channel name

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

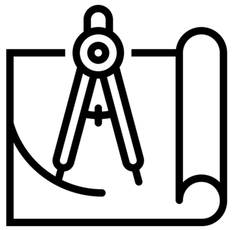
Task: retrieve all member emails from a Slack channel given the channel name

```
channel_name ⇒ {
  conversations_list()
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

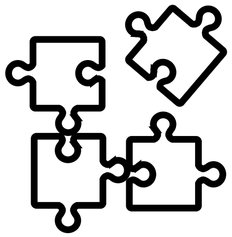
Can a synthesizer find this program?



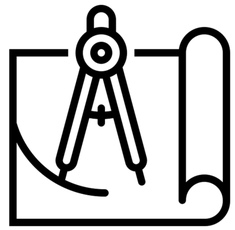
Specifications



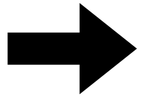
Specifications

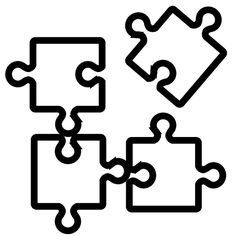


Components

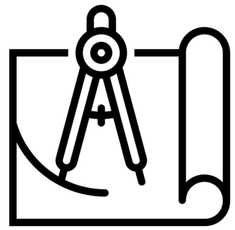


Specifications

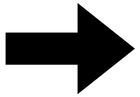


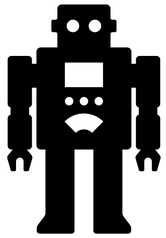


Components

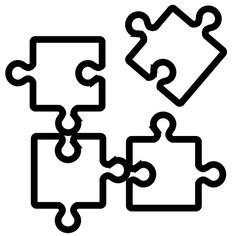


Specifications

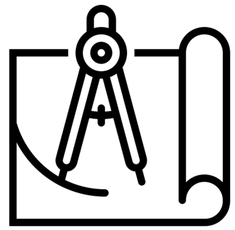




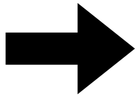
Synthesizer

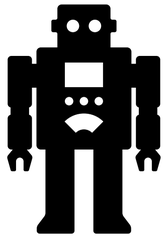


Components

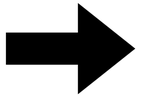


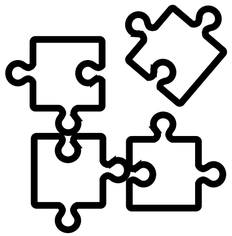
Specifications



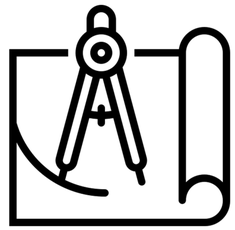


Synthesizer

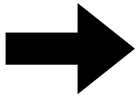


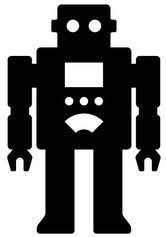


Components

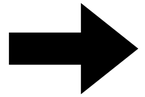


Specifications

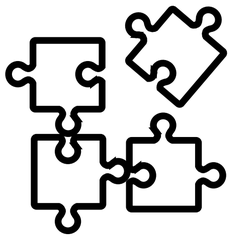




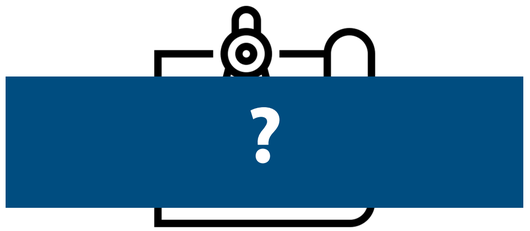
Synthesizer



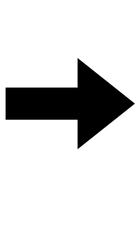
Programs



Components

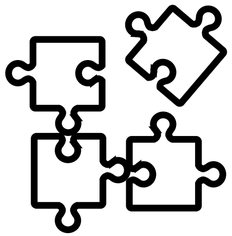


Specifications



Synthesizer

Programs



Components

I/O examples

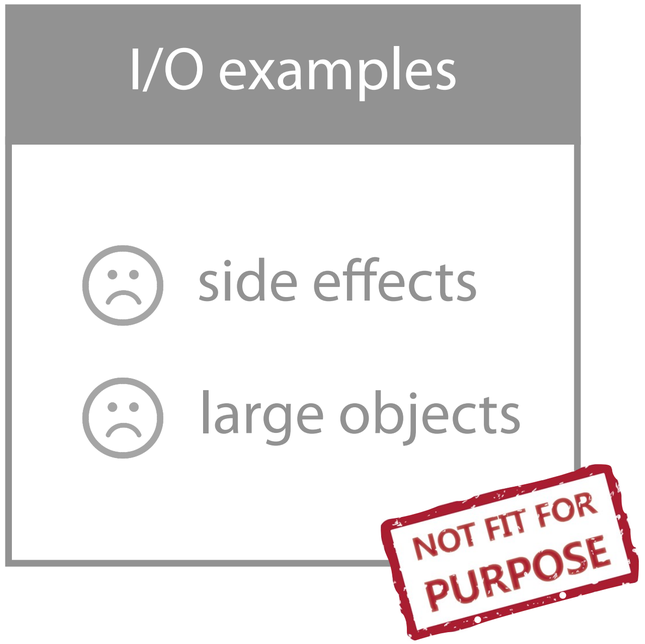
I/O examples

☹ side effects

I/O examples

☹ side effects

☹ large objects


I/O examples
side effects
large objects
NOT FIT FOR
PURPOSE

Specifications / **NL as Specifications?**

I/O examples

- side effects
- large objects

natural language

Specifications / **NL as Specifications?**

I/O examples

- side effects
- large objects

natural language

- too vague

Specifications / **NL as Specifications?**

I/O examples

- side effects
- large objects

natural language

- too vague

Specifications / **Types as Specifications?**

I/O examples

- side effects
- large objects

natural language

- too vague

types

Specifications / **Types as Specifications?**

I/O examples

- ☹ side effects
- ☹ large objects

natural language

- ☹ too vague

types

Specifications / **Types as Specifications?**

I/O examples	natural language	types
☹ side effects	☹ too vague	☹ coarse-grained
☹ large objects		


```
conversations_members :: String → [String]
```

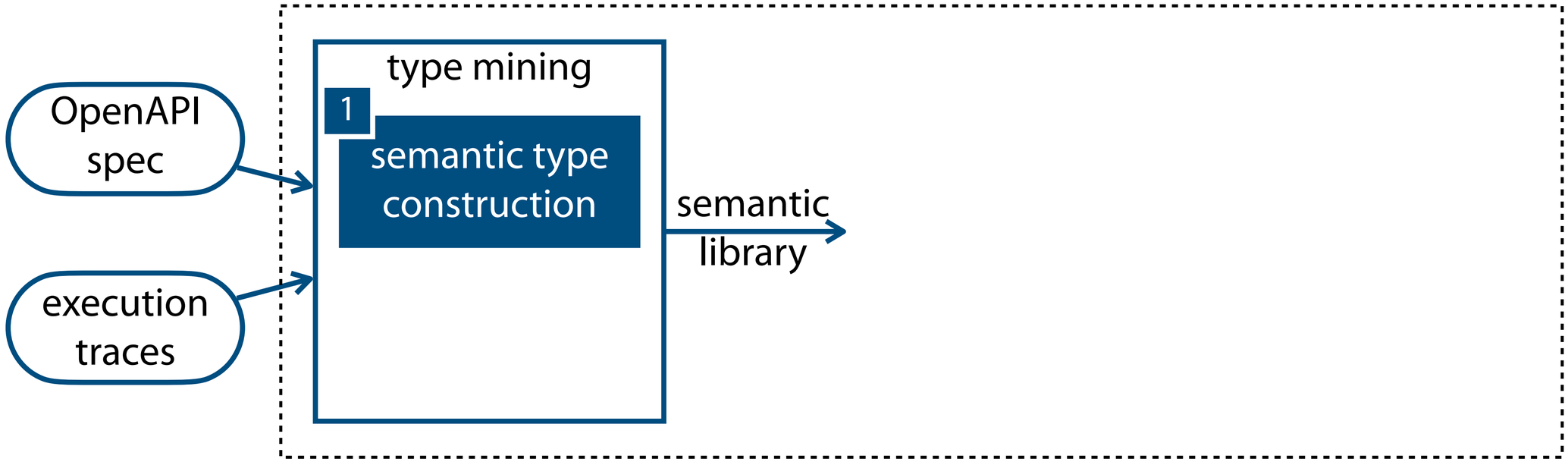
Our Contribution / **Semantic Types as Specifications!**

I/O examples	natural language	**semantic** types
☹ side effects	☹ too vague	☺ ~~coarse-grained~~
☹ large objects		

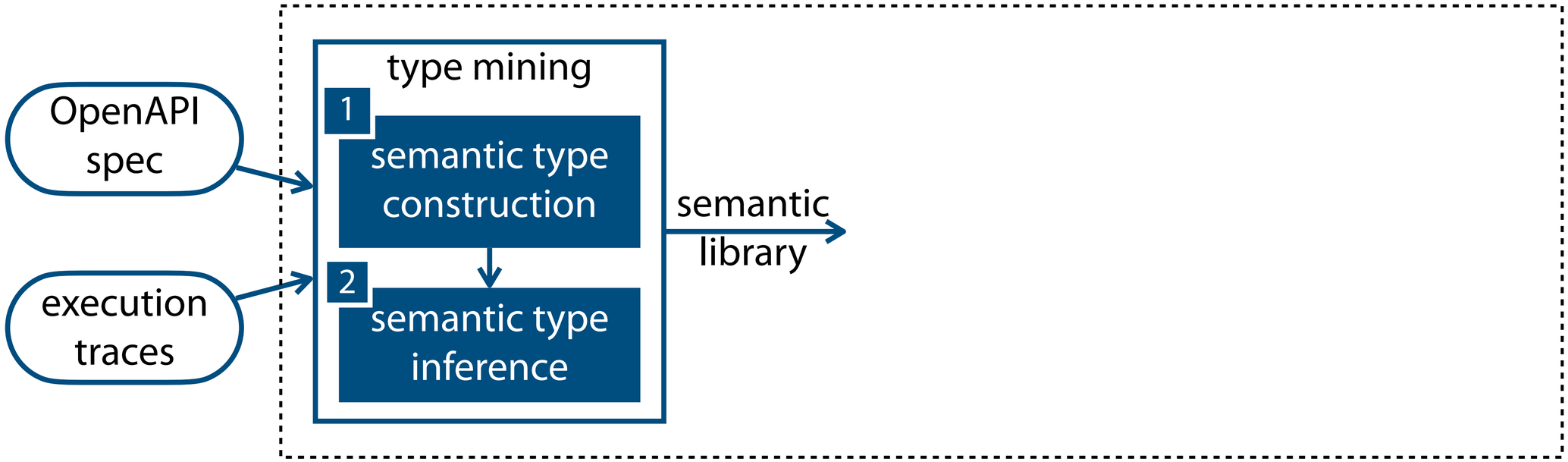
APIphany

A program synthesizer for REST APIs guided by semantic types

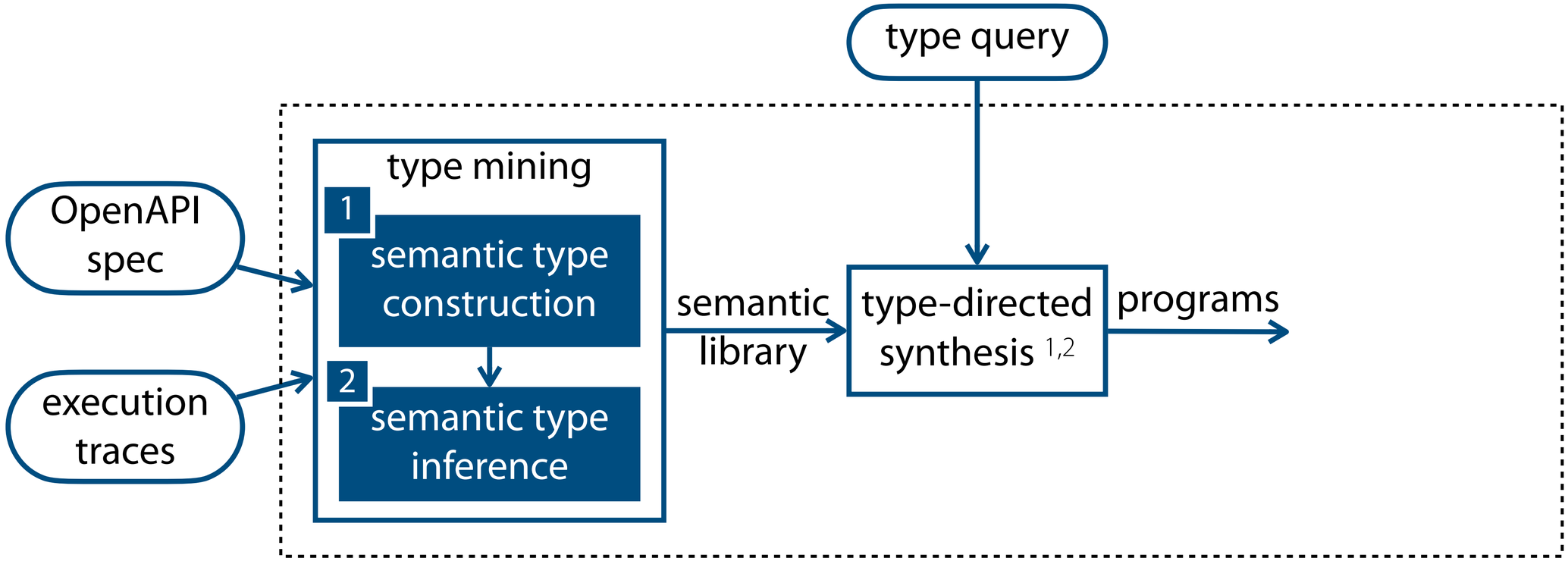
APIphany / **Architecture**



APIphany / **Architecture**

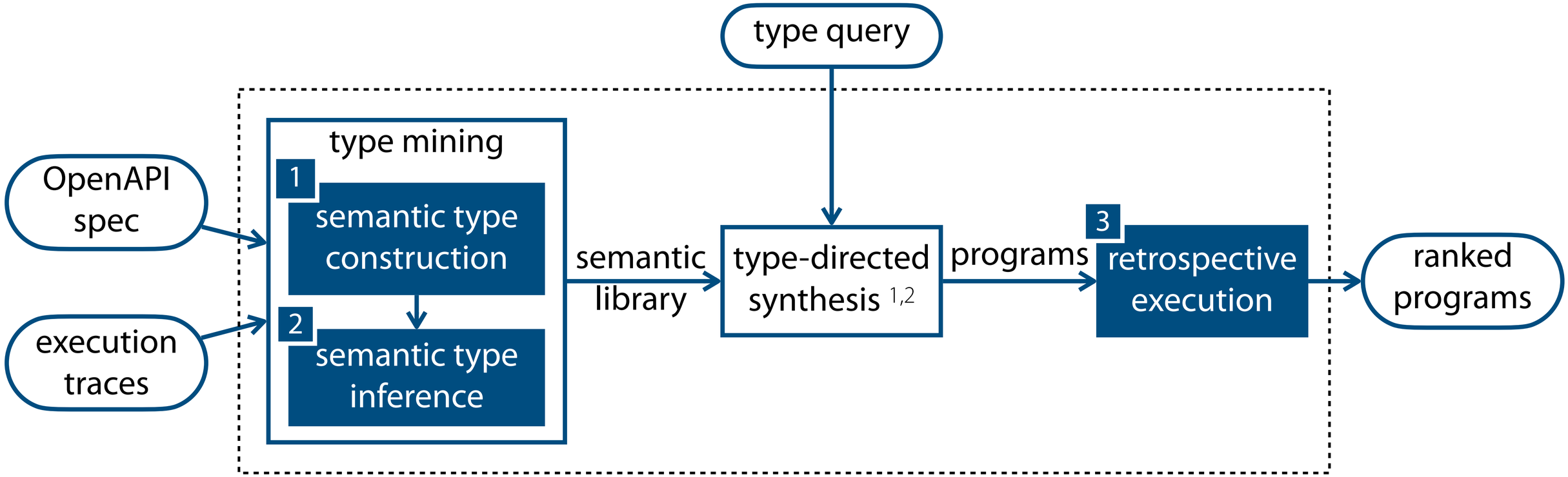


APIphany / **Architecture**



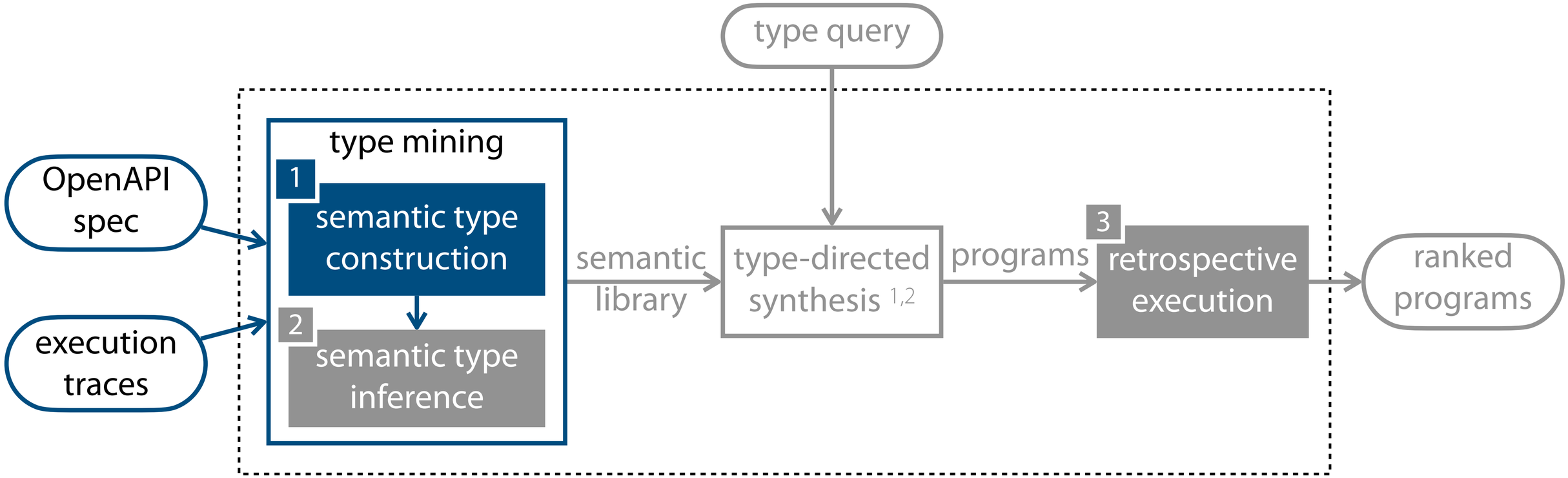
[1] Component-based synthesis for complex APIs. Feng et al. POPL'17
[2] Program synthesis by type-guided abstraction refinement. Guo et al. POPL'20

APIphany / **Architecture**



[1] Component-based synthesis for complex APIs. Feng et al. POPL'17
[2] Program synthesis by type-guided abstraction refinement. Guo et al. POPL'20

APIphany / **Architecture**



[1] Component-based synthesis for complex APIs. Feng et al. POPL'17
[2] Program synthesis by type-guided abstraction refinement. Guo et al. POPL'20

Task: retrieve all member emails from a Slack channel given the channel name

Type query: **?**

Task: retrieve all member emails from a Slack channel given the channel name

Type query: `“channel name” → [“user email”]`

Task: retrieve all member emails from a Slack channel given the channel name

Type query: `"channel name" → ["user email"]`

How to represent?

Task: retrieve all member emails from a Slack channel given the channel name

Type query: `"channel name" → ["user email"]`

Insight 1: object fields as types!

Objects

```
User    { id      :: String
        , profile :: Profile }

Profile { email   :: String
        , phone   :: String  }

Channel { creator :: String
        , name    :: String
        , id      :: String  }
```

Objects

```
User    { id       :: User.id
        , profile  :: Profile }

Profile { email    :: String
        , phone    :: String  }

Channel { creator  :: String
        , name     :: String
        , id       :: String  }
```

Objects

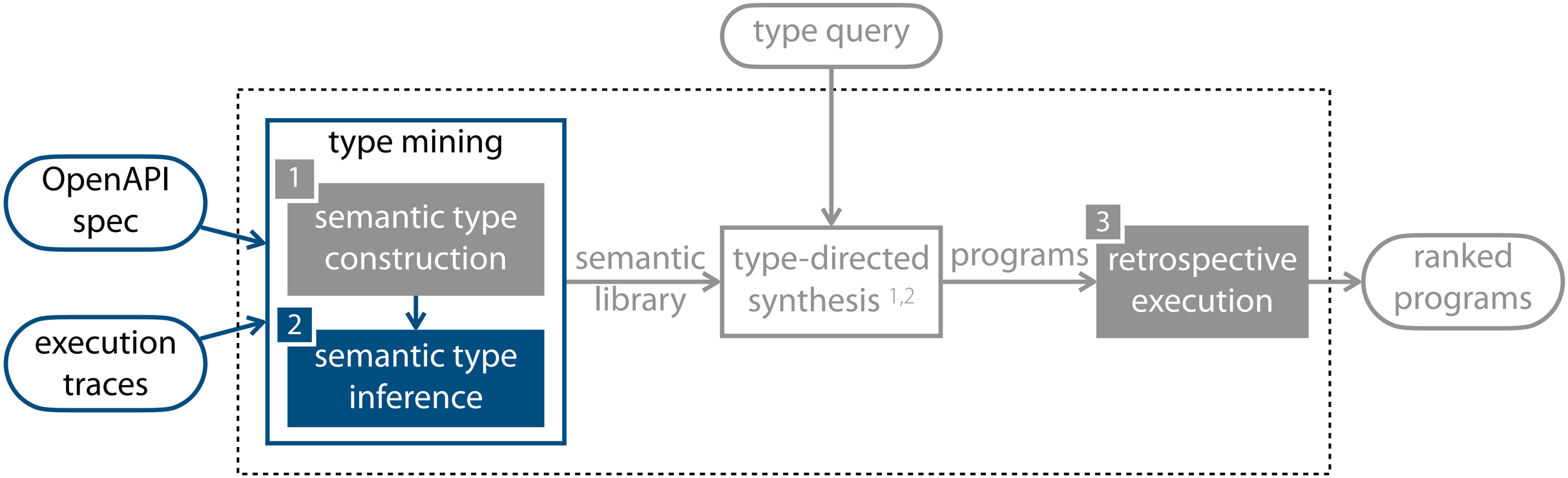
```
User    { id      :: User.id
        , profile :: Profile }

Profile { email   :: Profile.email
        , phone   :: Profile.phone

Channel { creator :: Channel.creator
        , name    :: Channel.name
        , id      :: Channel.id
```

Task: retrieve all member emails from a Slack channel given the channel name

Type query:

```
Channel.name → [Profile.email]
```

APIphany / **Architecture**



[1] Component-based synthesis for complex APIs. Feng et al. POPL'17
[2] Program synthesis by type-guided abstraction refinement. Guo et al. POPL'20

Objects

```
User    { id       :: User.id
        , profile  :: Profile }

Profile { phone    :: Profile.phone
        , email    :: Profile.email }

Channel { creator  :: Channel.creator
        , name     :: Channel.name
        , id       :: Channel.id }
```

Methods

```
users_info :: String → User

conversations_members :: String → [String]

conversations_list :: [Channel]
```

Objects

```
User    { id      :: User.id
        , email   :: Profile.email }

Channel { creator :: Channel.creator
        , name    :: Channel.name
        , id      :: Channel.id }
```

Methods

```
conversations_members :: String → [String]

conversations_list :: [Channel]
```

Insight 2: mine from traces!

Invocation

```
users_info(“UJ5RHEG4S”) ==
  { "id": “UJ5RHEG4S”
  , "name": "demo_user"
  , "profile":
    { “email": “xyz@gmail.com”
     …
    }
  }
```

Method Spec Type

```
users_info ::   String   → User
```

Invocation

```
users_info("UJ5RHEG4S") =
```

```
{ "id": "UJ5RHEG4S"
, "name": "demo_user"
, "profile":
  { "email": "xyz@gmail.com"
   …
  }
}
```

Method Spec Type

```
users_info :: String → User
```

Invocation

```
users_info("UJ5RHEG4S") ==
  { "id": "UJ5RHEG4S"
  , "name": "demo_user"
  , "profile":
    { "email": "xyz@gmail.com"
     …
    }
  }
```

User.id

Method Spec Type

```
users_info :: String → User
```

Invocation

```
users_info("UJ5RHEG4S")
```

User.id

```
{ "id": "UJ5RHEG4S"
, "name": "demo_user"
, "profile":
  { "email": "xyz@gmail.com"
    …
  }
}
```

Method Spec Type

```
users_info :: String → User
```

Invocation

```
users_info("UJ5RHEG4S") =
  { "id": "UJ5RHEG4S"
  , "name": "demo_user"
  , "profile":
    { "email": "xyz@gmail.com"
     …
    }
  }
```

Method Spec Type

```
users_info :: User.id → User
```

Objects

```
User    { id      :: User.id
        , profile :: Profile }

Profile { phone   :: Profile.phone
        , email   :: Profile.email }

Channel { creator :: Channel.creator
        , name    :: Channel.name
        , id      :: Channel.id }
```

Methods

```
users_info   :: User.id → User
convs_members :: Channel.id → [ User.id ]

convs_list :: [Channel]
```

Objects

```
User    { id       :: User.id
        , email    :: Profile.email }

Channel { creator  :: Channel.creator
        , name     :: Channel.name
        , id       :: Channel.id }
```

Methods

```
convs_members :: Channel.id → [ User.id ]

convs_list :: [Channel]
```

Are we done?

Objects

```
User    { id       :: User.id
        , profile  :: Profile }

Profile { phone    :: Profile.phone
        , email    :: Profile.email }

Channel { creator  :: Channel.creator
        , name     :: Channel.name
        , id       :: Channel.id }
```

Methods

```
users_info ::    User.id → User

convs_members ::  Channel.id  → [  User.id  ]

convs_list :: [Channel]
```

A user ID

Objects

```
User    { id       :: User.id
        , profile  :: Profile }

Profile { phone    :: Profile.phone
        , email    :: Profile.email }

Channel { creator  :: Channel.creator
        , name     :: Channel.name
        , id       :: Channel.id }
```

Methods

```
users_info ::    User.id → User

convs_members ::  Channel.id  → [  User.id  ]

convs_list :: [Channel]
```

A user ID

Traces!

Invocation

```
{ "id": "C123"
, "name": “general"
, "creator": "UJ5RHEG4S"
, ...
}
```

Object Spec Type

```
Channel { creator :: Channel.creator
        , name    :: Channel.name
        , id      :: Channel.id }
```

Invocation

```
{ "id": "C123"
, "name": “general"
, "creator": "UJ5RHEG4S"
, ...
}
```

User.id

Object Spec Type

```
Channel { creator :: Channel.creator
        , name    :: Channel.name
        , id      :: Channel.id }
```

Invocation

```
{ "id": "C123"
, "name": “general"
, "creator": "UJ5RHEG4S"
, ...
}
```

Object Spec Type

```
Channel { creator :: User.id
        , name    :: Channel.name
        , id      :: Channel.id }
```

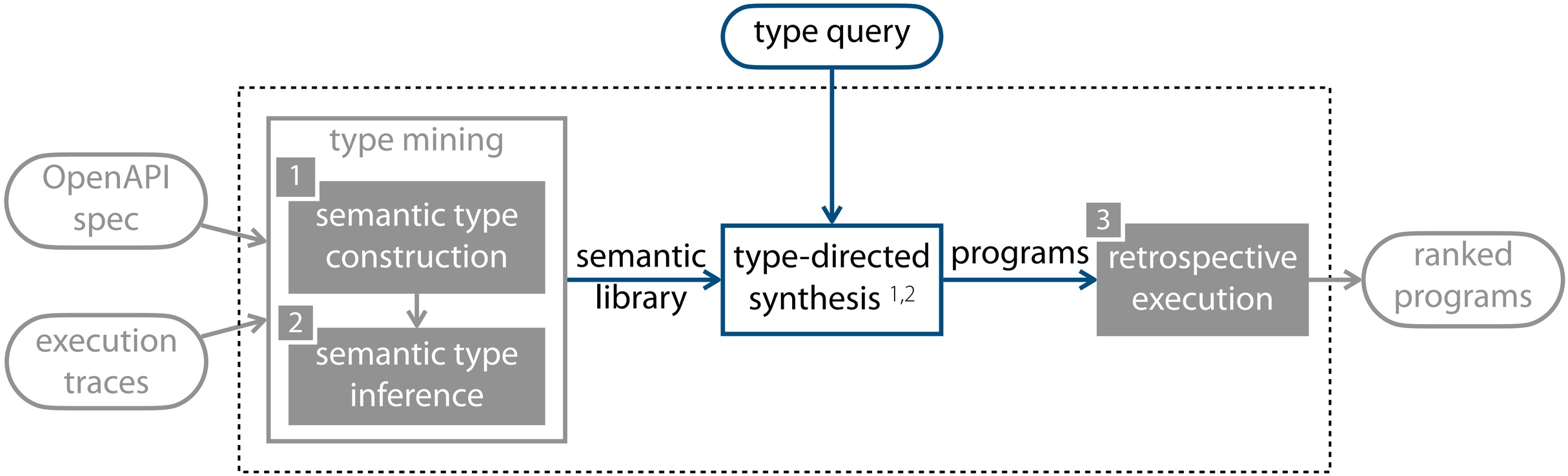
Objects

```
User    { id      :: User.id
        , profile :: Profile }

Profile { phone   :: Profile.phone
        , email   :: Profile.email }

Channel { creator :: User.id
        , name    :: Channel.name
        , id      :: Channel.id }
```

Methods

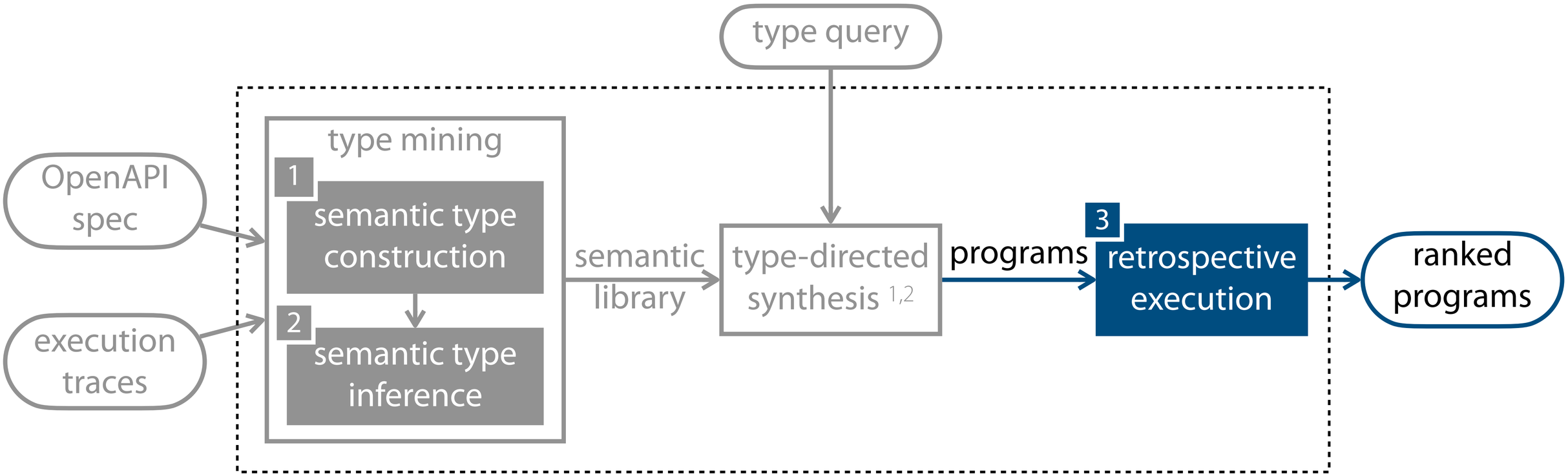
```
users_info   :: User.id → User

convs_members :: Channel.id → [ User.id ]

convs_list :: [Channel]
```

APIphany / **Architecture**



[1] Component-based synthesis for complex APIs. Feng et al. POPL'17
[2] Program synthesis by type-guided abstraction refinement. Guo et al. POPL'20

APIphany / **Architecture**



Type-Directed Synthesis / **Program Ranking**

```
Channel.Name → [Profile.Email]
```

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

Desired Solution

Type-Directed Synthesis / **Program Ranking**

`Channel.Name → [Profile.Email]`

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

Desired Solution

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { let uid = c.creator
              let u = users_info(user=uid)
              return u.profile.email })
  })}
```

Candidate #1

Type-Directed Synthesis / **Program Ranking**

`Channel.Name → [Profile.Email]`

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

Desired Solution

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { let uid = c.creator
              let u = users_info(user=uid)
              return u.profile.email })
  })}
```

Always returns a singleton array

Type-Directed Synthesis / **Program Ranking**

`Channel.Name → [Profile.Email]`

```
channel_name ⇒ {
  conversations_list()
    .filter(c ⇒ c.name == channel_name)
    .map(c ⇒ { conversations_members(c.id)
      .map(uid ⇒ { let u = users_info(user=uid)
                   return u.profile.email })
  })}
```

Desired Solution

```
channel_name ⇒ {
  conversations_open()
    .filter(c ⇒ c.name == channel_name)
    .map(c ⇒ { conversations_members(c.id)
      .map(uid ⇒ { let u = users_info(user=uid)
                   return u.profile.email })
  })}
```

Candidate #2

Type-Directed Synthesis / **Program Ranking**

`Channel.Name → [Profile.Email]`

```
channel_name ⇒ {
  conversations_list()
    .filter(c ⇒ c.name == channel_name)
    .map(c ⇒ { conversations_members(c.id)
      .map(uid ⇒ { let u = users_info(user=uid)
                   return u.profile.email })
  })}
```

Desired Solution

```
channel_name ⇒ {
  conversations_open()
    .filter(c ⇒ c.name == channel_name)
    .map(c ⇒ { conversations_members(c.id)
      .map(uid ⇒ { let u = users_info(user=uid)
                   return u.profile.email })
  })}
```

Always fails

Type-Directed Synthesis / **Program Ranking**

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

Desired Solution

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { let uid = c.creator
              let u = users_info(user=uid)
              return u.profile.email })
  })}
```

Candidate #1

```
channel_name ⇒ {
  conversations_open()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

Candidate #2

Type-Directed Synthesis / **Program Ranking**

```
channel_name ⇒ {
  conversations_list()
                    return u.profile.email })
  })}
```

Desired Solution

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { let uid = c.creator
              let u = users_info(user=uid)
              return u.profile.email })
  })}
```

Execute programs?

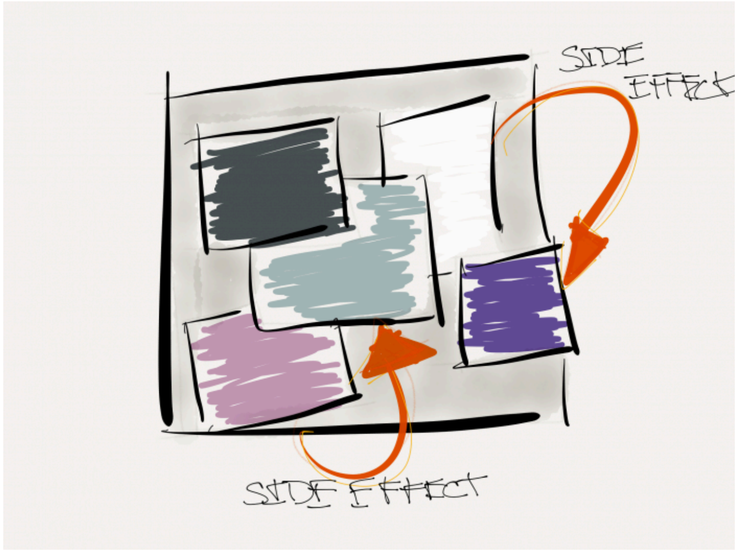
```
channel_name ⇒ {
  conversations_open()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

Candidate #2

Service providers set a rate limit

Program Ranking / **Challenges in REST API Execution**

Service providers set a rate limit



Many API calls have side effects

Program Ranking / **Retrospective Execution**

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

Desired Solution

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { let uid = c.creator
              let u = users_info(user=uid)
              return u.profile.email })
  })}
```

Candidate #1

```
channel_name ⇒ {
  conversations_open()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

Candidate #2

Program Ranking / **Retrospective Execution**

```
channel_name ⇒ {
  conversations_list()
                    return u.profile.email })
  })}
```

Desired Solution

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { let uid = c.creator
              let u = users_info(user=uid)
              return u.profile.email })
  })}
```

Insight 3: replay from execution traces!

```
channel_name ⇒ {
  conversations_open()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

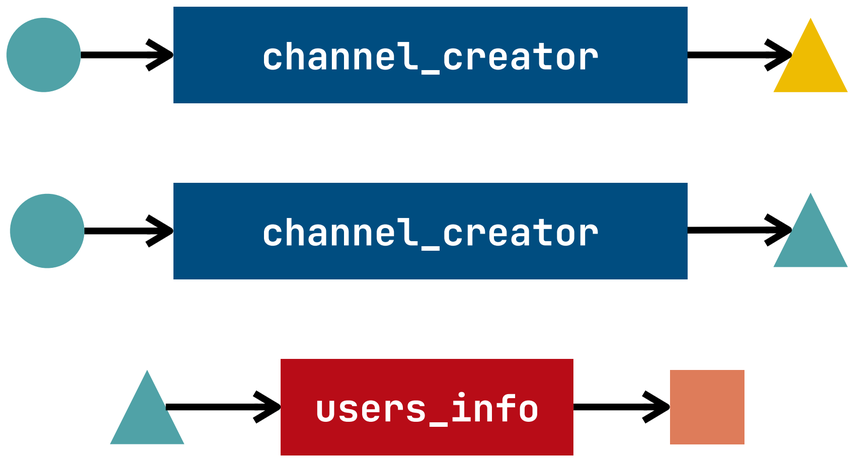
Candidate #2

Execution traces

Program

```
...
  let uid = channel_creator(c)
  let u = users_info(user=uid)
...
```

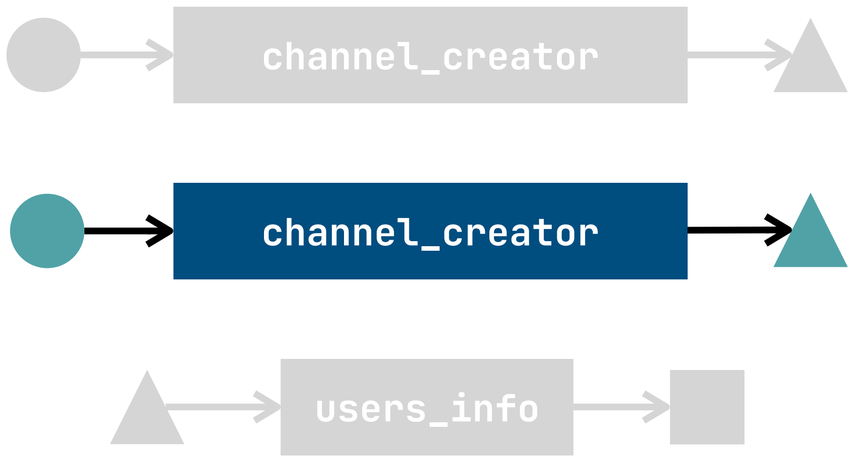
Execution traces



Program

```
...
  let uid = channel_creator(c)
  let u = users_info(user=uid)
...
```

Execution traces

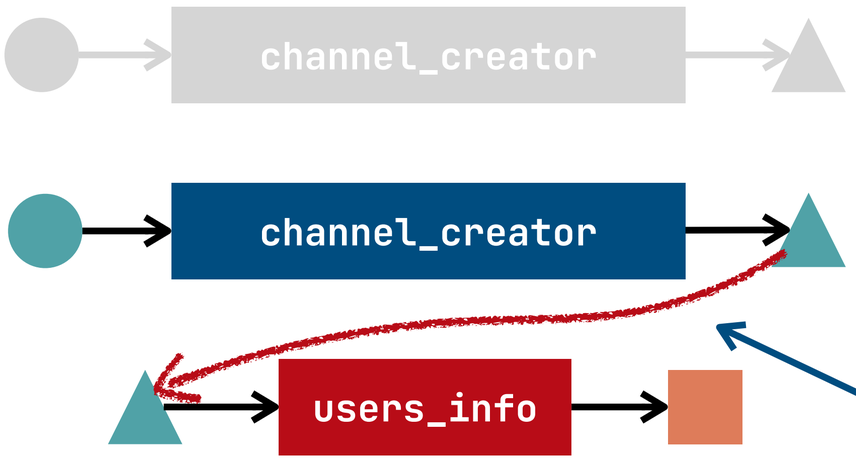


Program

```
...
  let uid = channel_creator(c)
  let u = users_info(user=uid)
...
```

Program Ranking / **Retrospective Execution**

Execution traces

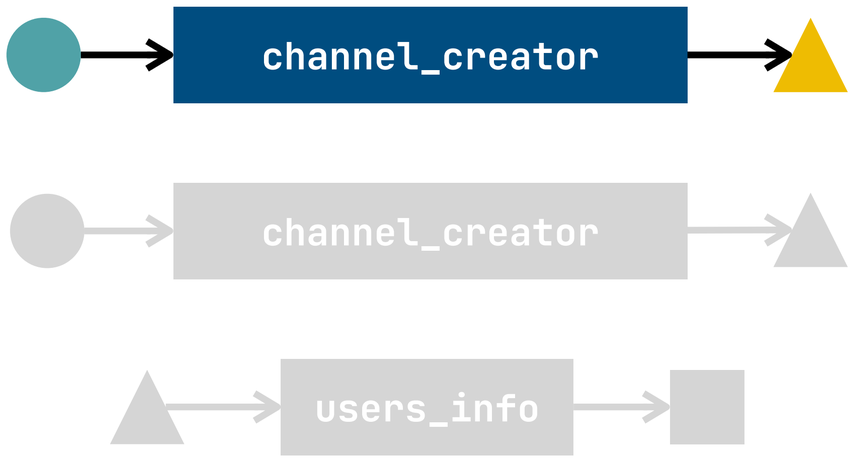


Program

```
...
  let uid = channel_creator(c)
  let u = users_info(user=uid)
...
```

exact execution

Execution traces

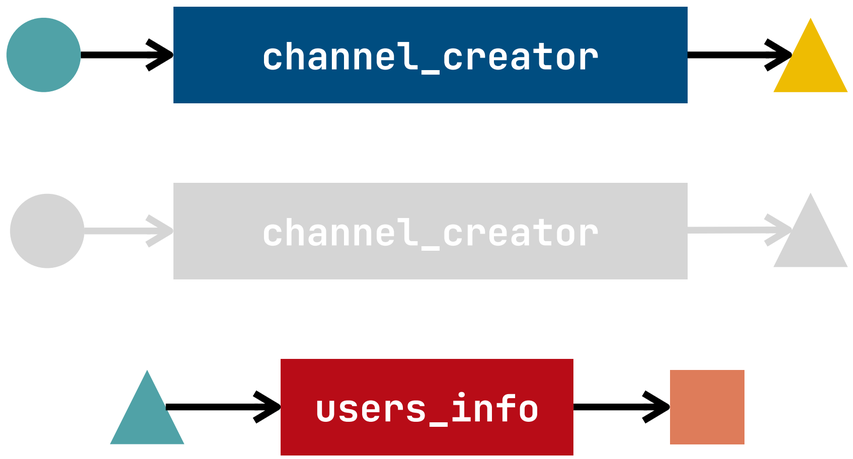


Program

```
...
  let uid = channel_creator(c)
  let u = users_info(user=uid)
...
```

Program Ranking / **Retrospective Execution**

Execution traces

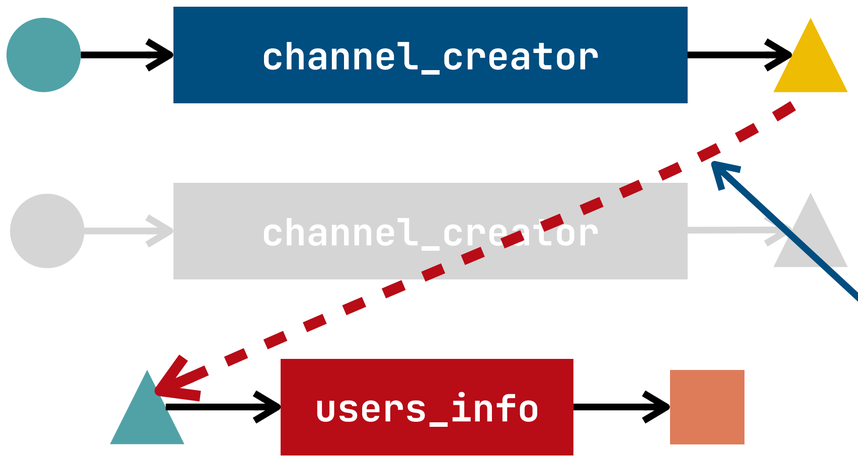


Program

```
...
  let uid = channel_creator(c)
  let u = users_info(user=uid)
...
```

Program Ranking / **Retrospective Execution**

Execution traces



Program

```
...
  let uid = channel_creator(c)
  let u = users_info(user=uid)
...
```

approximate execution

Type-Directed Synthesis / **Program Ranking**

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

score:50

Desired Solution

```
channel_name ⇒ {
  conversations_list()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { let uid = c.creator
              let u = users_info(user=uid)
              return u.profile.email })
  })}
```

score:5

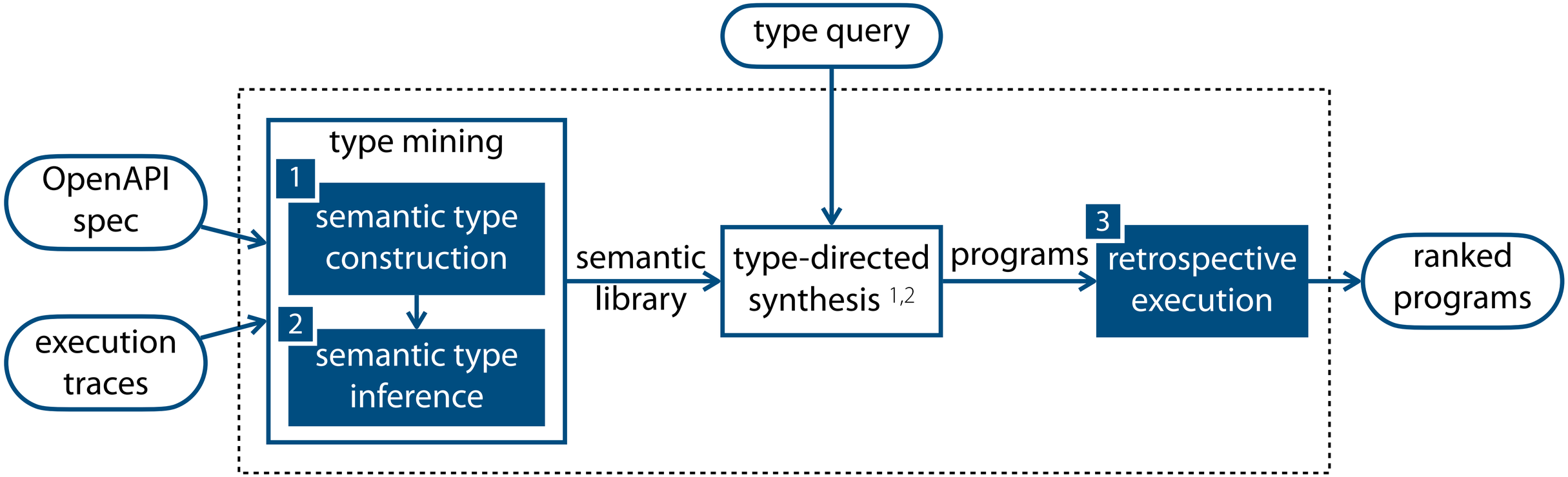
Candidate #1

```
channel_name ⇒ {
  conversations_open()
   .filter(c ⇒ c.name == channel_name)
   .map(c ⇒ { conversations_members(c.id)
     .map(uid ⇒ { let u = users_info(user=uid)
                  return u.profile.email })
  })}
```

score:-1

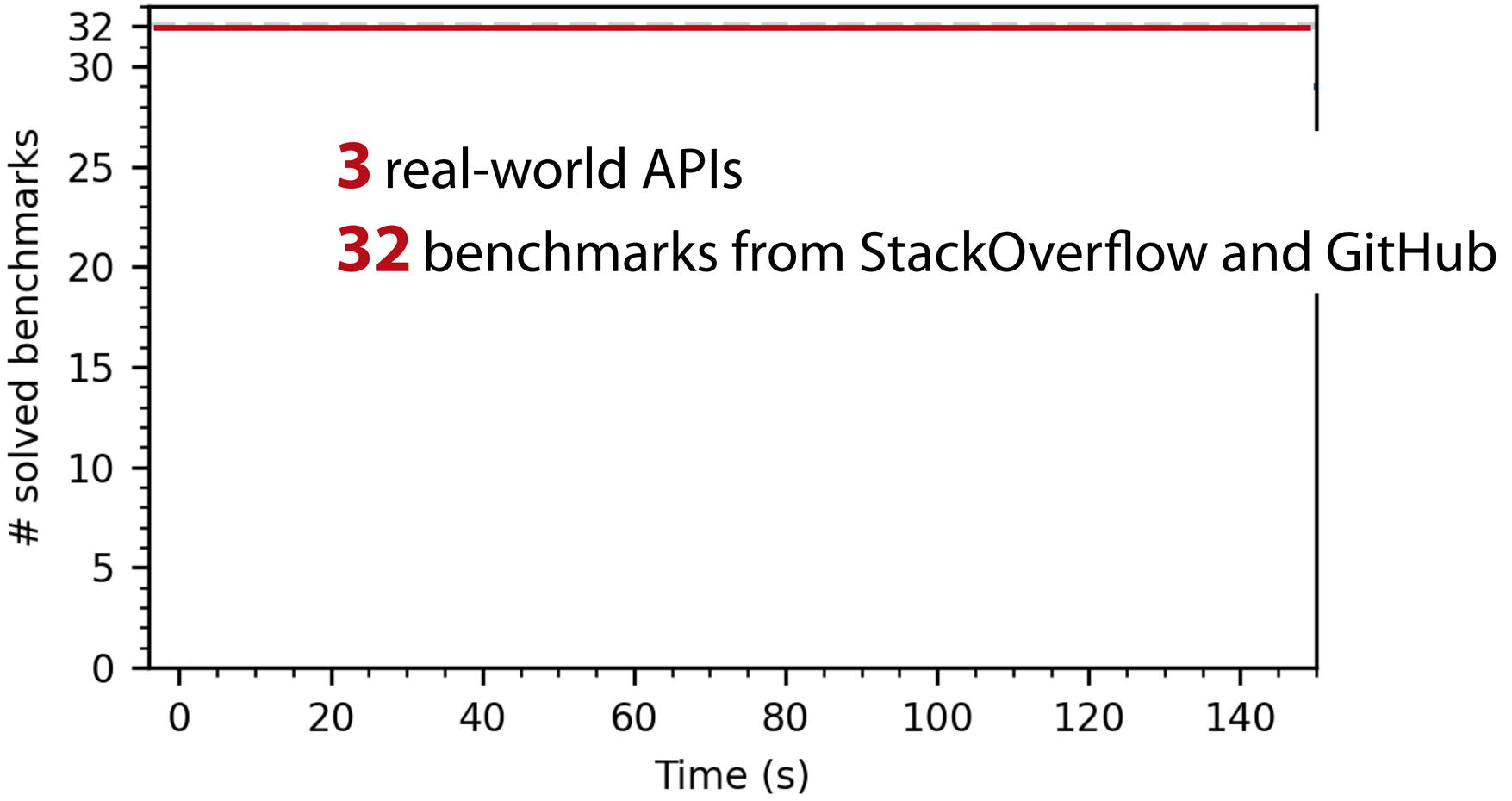
Candidate #2

APIphany / **Architecture**

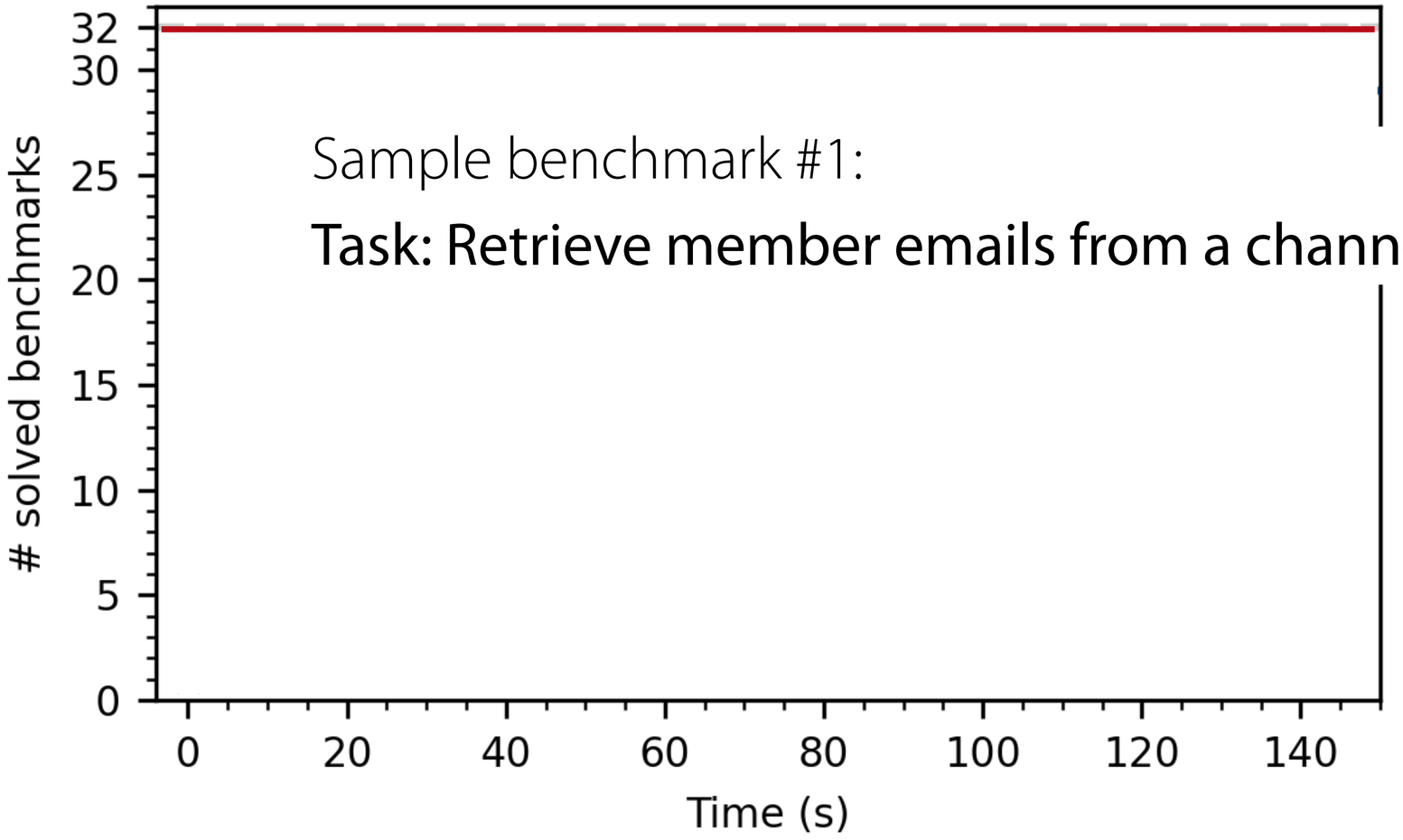


[1] Component-based synthesis for complex APIs. Feng et al. POPL'17
[2] Program synthesis by type-guided abstraction refinement. Guo et al. POPL'20

Evaluation / **Solved Benchmarks**



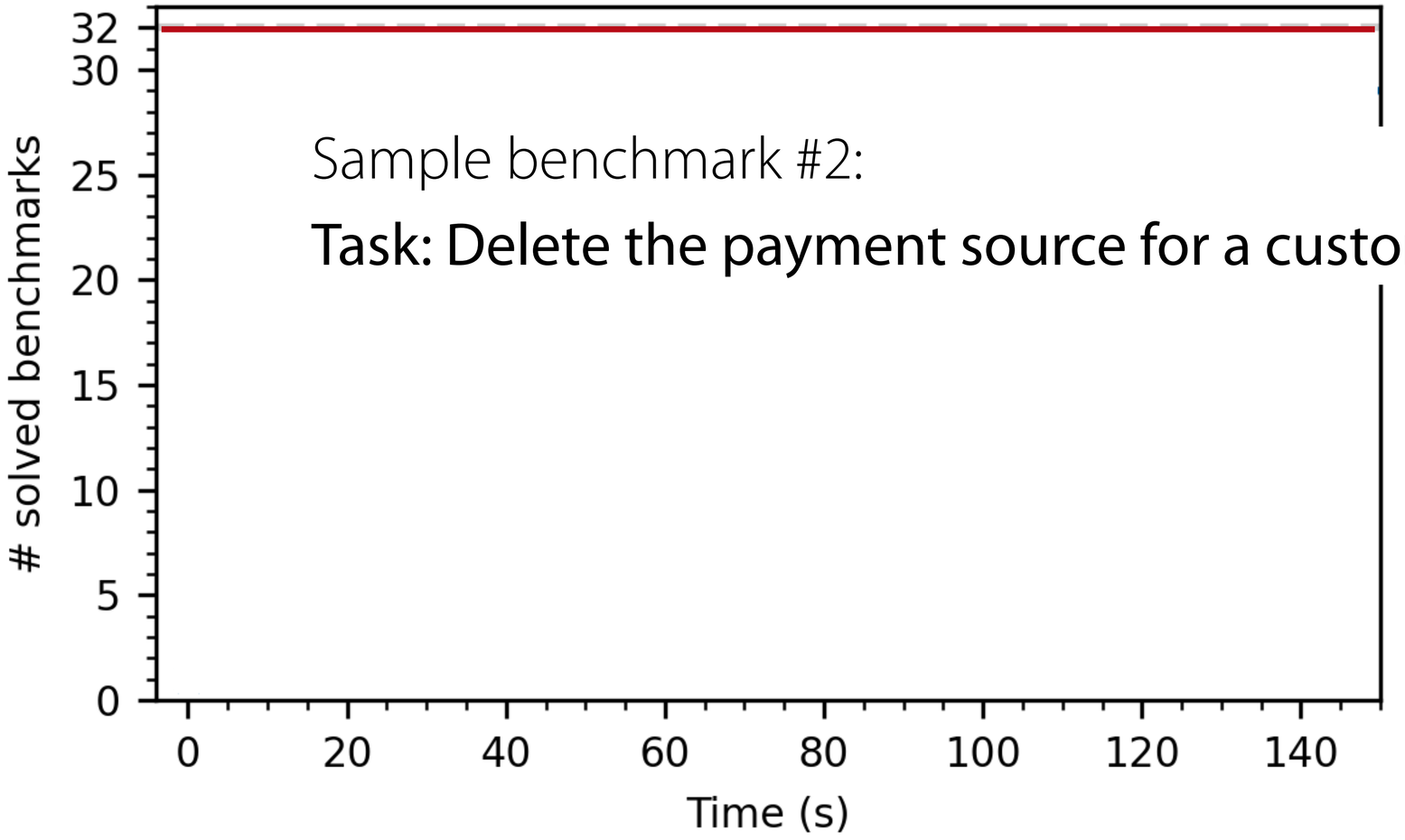
Evaluation / **Solved Benchmarks**



Sample benchmark #1:

Task: Retrieve member emails from a channel name

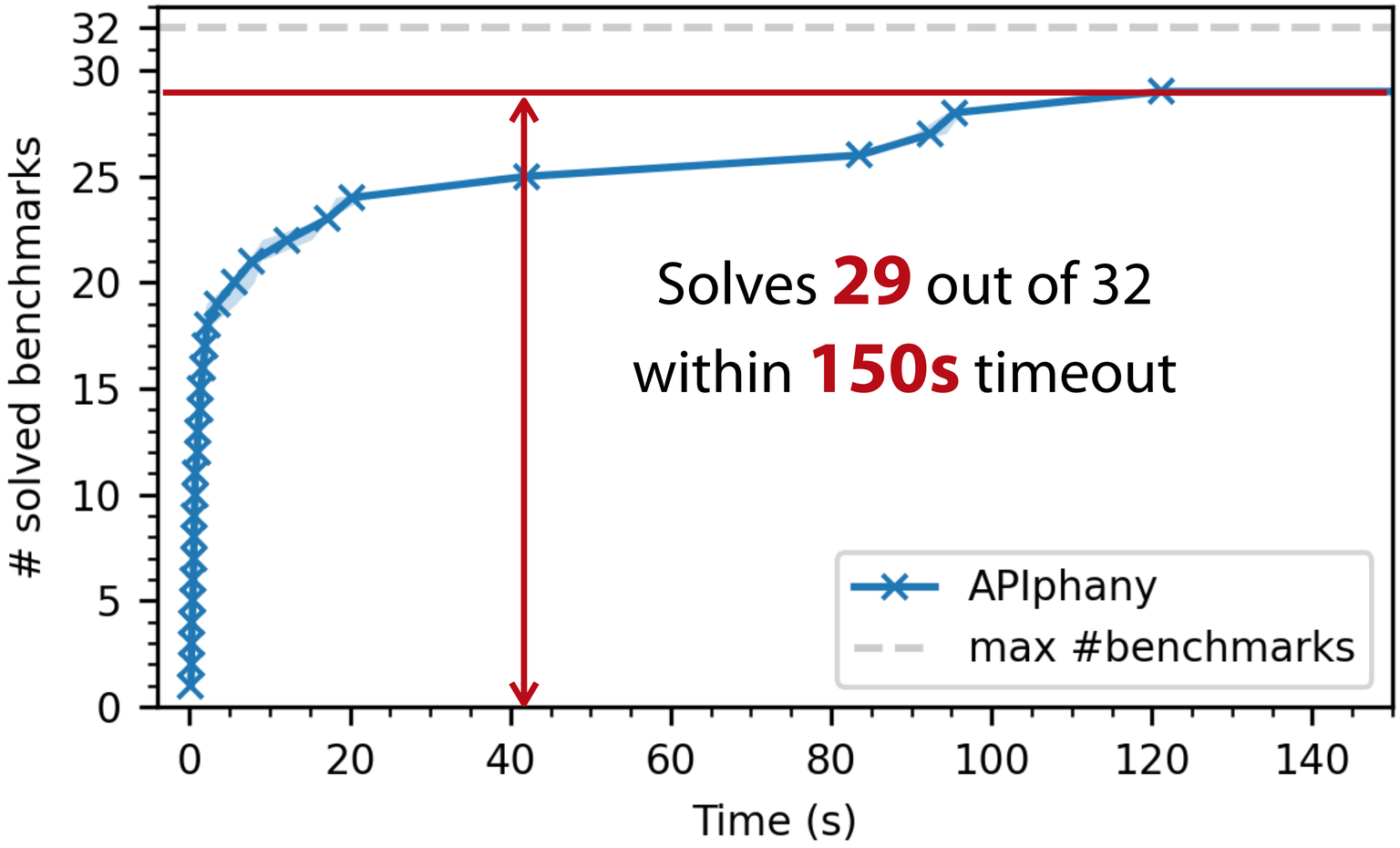
Evaluation / **Solved Benchmarks**



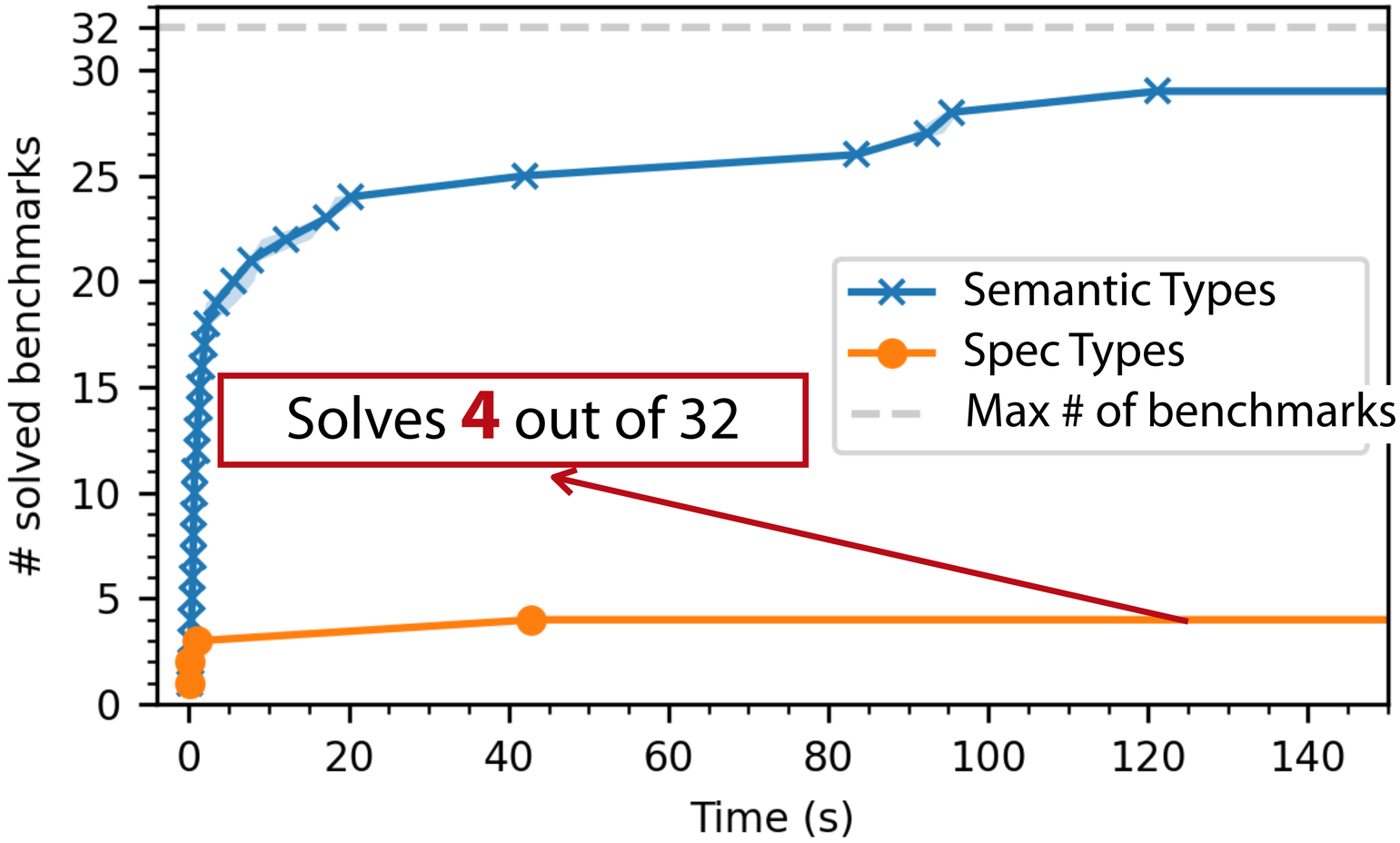
Sample benchmark #2:

Task: Delete the payment source for a customer

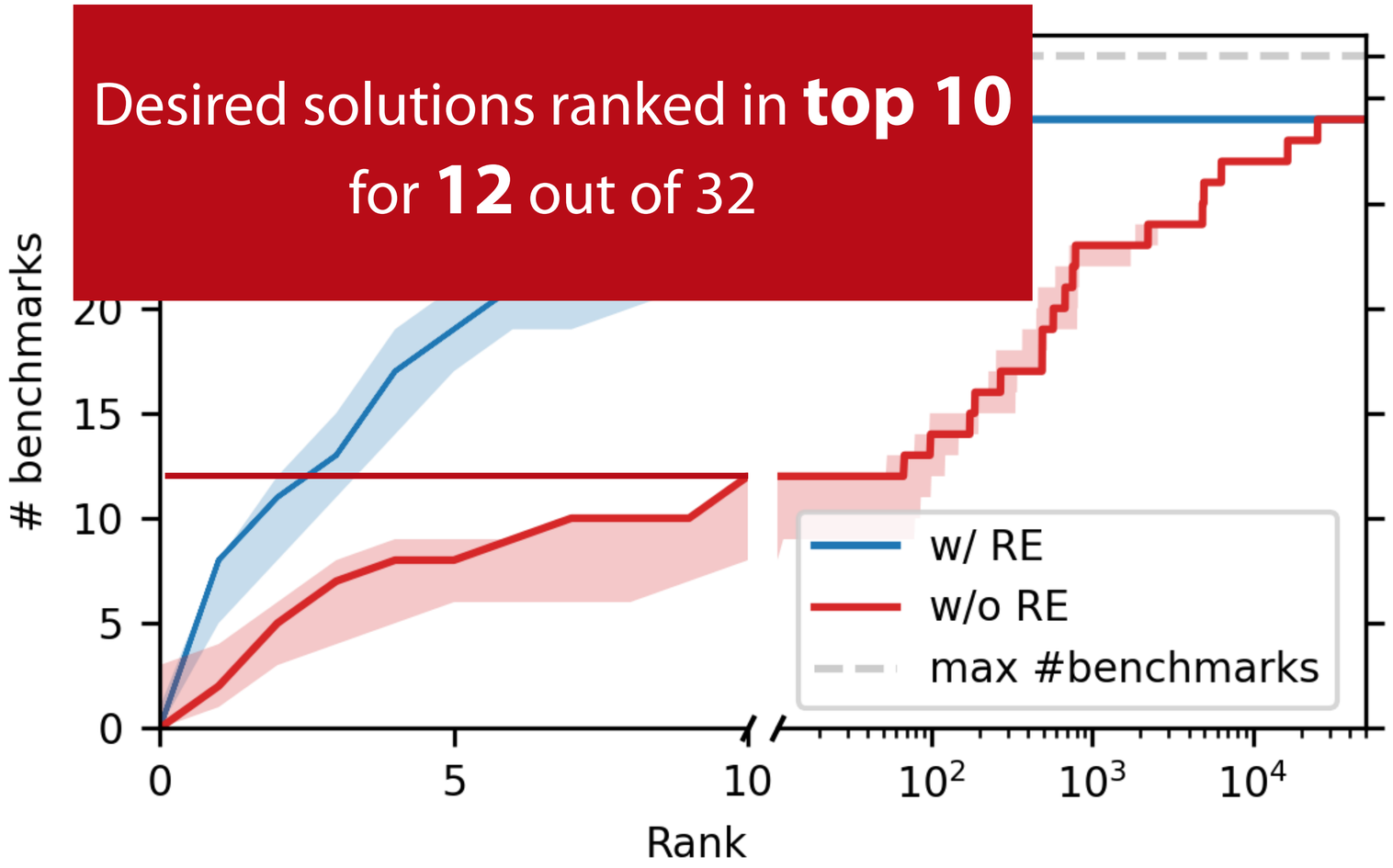
Evaluation / **Solved Benchmarks**



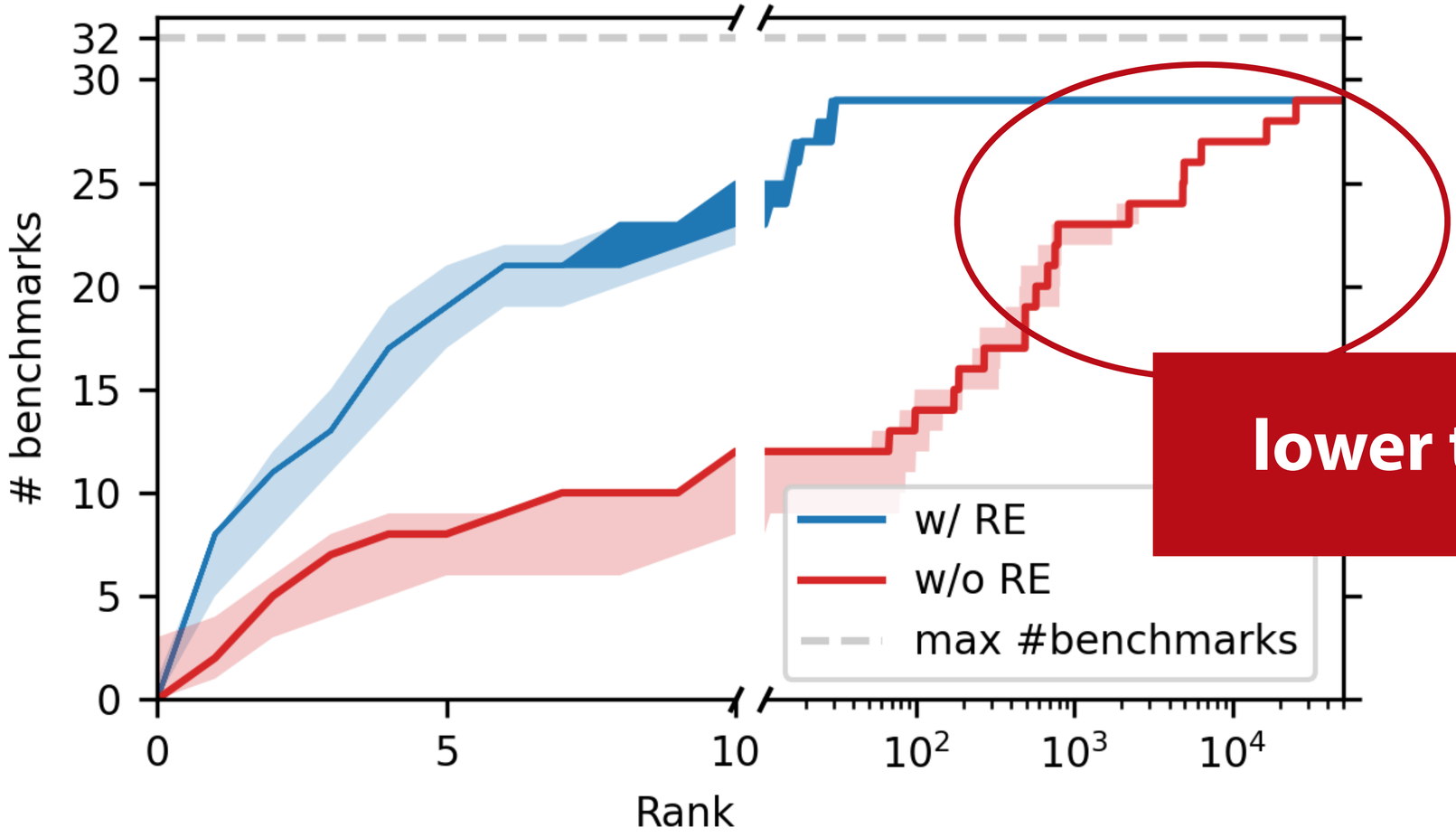
Evaluation / **Spec vs Semantic Types**



Evaluation / **RE vs No RE**

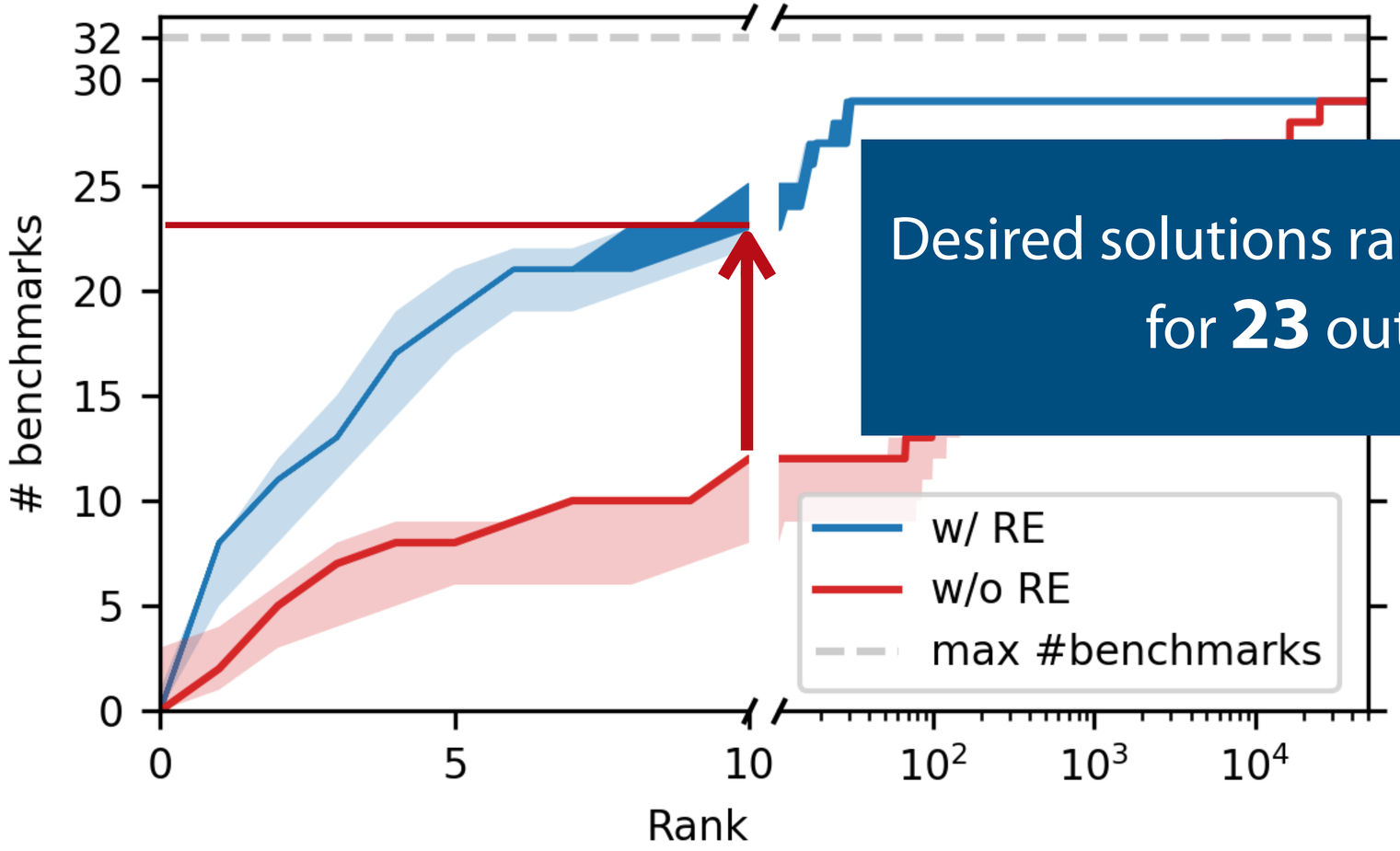


Evaluation / **RE vs No RE**



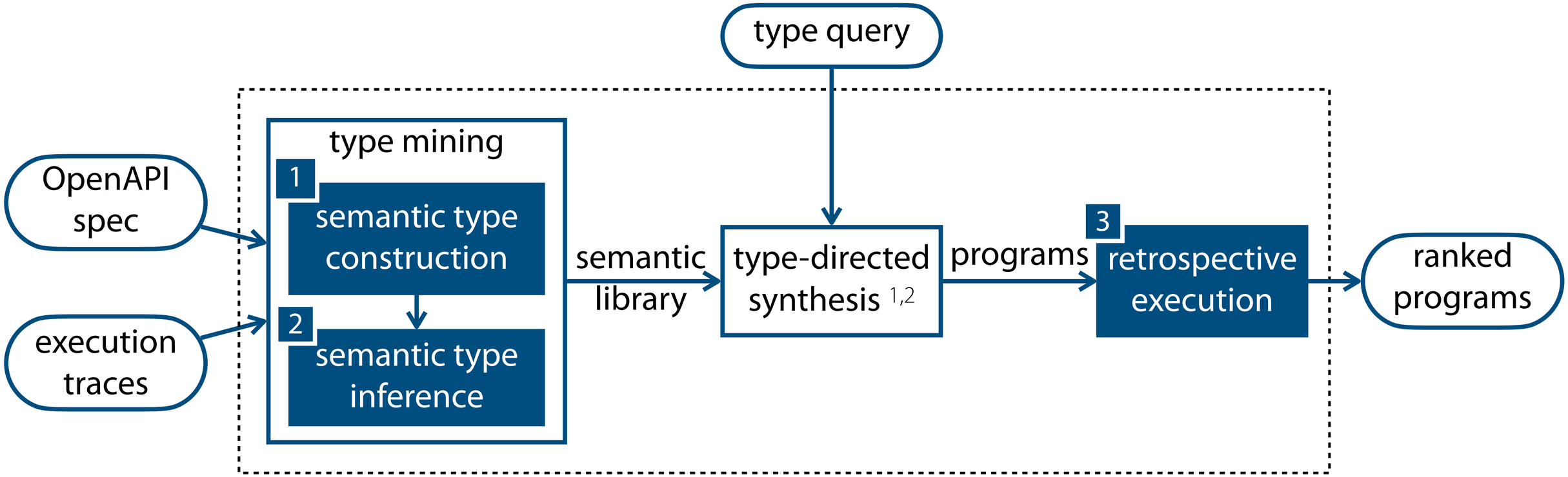
lower than 1000

Evaluation / **RE vs No RE**



Desired solutions ranked in **top 10** for **23** out of 32

APIphany / **Architecture**



[1] Component-based synthesis for complex APIs. Feng et al. POPL'17
[2] Program synthesis by type-guided abstraction refinement. Guo et al. POPL'20