#31: Paper Discussion and Synthesis with Abstract Interpretation (cont.)

Sankha Narayan Guria

EECS 700: Introduction to Program Synthesis

Logistics

- Last paper reading assignment (to be released today)

- Start working on your project
  - Meet with me if you need to discuss anything
- Deliverables:
  - Final report (5-8 pages): intro, describe your algorithm with an example, general algorithm, design decisions, evaluations
  - GitHub repo link (put in your report)
  - Presentations (13 mins each)

Regae

[Zhang, Lowmanstone, Wang, Glassman, UIST'20]

Better UI for a regex synthesizer

Regae: contributions

- Novel way to express intent: semantics augmentation
- Novel way to explain synthesis results to user: data augmentation
- Automata-theoretic algorithms to generate explanatory examples
  - familiar examples with different output, corner cases, distinguishing examples
- Usability confirmed by user study
  - Completion rate: 12/12 vs 4/12; twice more confident; less cognitive load

Regae: limitations

Limited to regexes
    Which parts are generalizable and which not?
Marking as general affects completeness
Not tolerant to user mistakes
User study participants might not be representative

Regae: questions

- Behavioral constraints? Structural constraints? Search strategy?
  - IO examples
  - Built-in DSL
  - Top-down enumerative search

Regae: questions

- Does semantic augmentation contribute to behavioral or structural constraints, or something else?
  - Structural because it affects the search space
- What about data augmentation?
  - Directly contributes only to result comprehension
  - Indirectly to behavioral because users can use those examples as input

Regae: questions

- When can we soundly reject the sketch concat(<num>, e)?
  - If e.g. <num> is marked excluded [that's not what I meant]
  - When there is a positive example that doesn't start with a number
  - More generally, replace e with star(<any>) and check whether all positives can be parsed!
    - if under not, then replace with an empty-language regex
  - Another idea is define equivalence on regexes, e.g. optional(star(e)) is equivalent to star(e)

Regae: questions

- Why is it important to randomize the order of control vs treatment?

RbSyn

[Guria, Foster, Van Horn, PLDI'21]

Program synthesis from side effects

RbSyn: contributions

- Using side-effects to guide search
- Rule based merging to create if-then-else branches
- Automatic side effect inference from test failures
- Evaluated on programs from real-world benchmarks

RbSyn: limitations

- Cannot synthesize loops/lambdas/etc.
- Effect annotation burden may require domain insight
- Limited to typed subset of Ruby

RbSyn: questions

- Branch merging strategy wrt Synquid and EUSolver?
  - Synquid uses liquid abduction
  - EUSolver uses decision tree learning (information gain heuristic)
  - RbSyn uses rules-based approach as no counterexample possible

RbSyn: questions

- Scaling of branch merging with changing no. of tests (N)
  - N! ways of merges to be checked
  - It is M! if there are only M distinct programs from tests
  - The bottleneck is the number of distinct solutions to individual tests

RbSyn: questions

- Order of search for effect annotations:
  - `User.name`
  - `User`
  - `*`

  - Methods to be considered in this order: `name=`, `save`, `delete`

RbSyn: questions

- Why does RbSyn stay sound?
  - Wrong effect inferred: Adds program to the work list, no sound program is eliminated from the work list
  - Wrong method substituted: All correct effect annotated method choices are enumerated

Today

- Synthesizing data-structure manipulation from storyboards
  - Rishabh Singh, Armando Solar-Lezama

- **Absynthe: Abstract Interpretation-Guided Synthesis**
  - Sankha Narayan Guria, Jeff Foster, David Van Horn

Example

arg0

	id	valueA
0	255	1141
1	91	1130
2	347	830
⋮	⋮	⋮
8	225	638
9	257	616

arg1

	id	valueB
0	255	1231
1	91	1170
2	5247	954
⋮	⋮	⋮
12	211	575
13	25	530

arg2

`"valueA ≠ valueB"`

	id	valueA	valueB
0	255	1141	1231
1	91	1130	1170
2	347	830	870
5	159	715	734
8	225	638	644

Types and column labels are a potential good abstraction

`{"id", "valueA", "valueB"} x DataFrame`

Types Abstract Interpreter

```
class PyTypeInterp
```

Parameter to Absynthe
for a class of problems

Pandas data frame merge

```
# left.merge(right, opts)
df1.merge(df2, on = ['id'])
```

```
end
```

Types Abstract Interpreter

```
class PyTypeInterp
  def self.pd_merge(left, right, opt)
    if left  ⊆ DataFrame &&
       right ⊆ DataFrame &&
       opt   ⊆ { on: Array<String>}
       DataFrame
    end
  end
```

Pandas data frame query

```
# df.query(pred)
df.query('valueA > 10')
```

```
end
```

Columns Abstract Interpreter

```
class ColNameInterp



end
```

Pandas data frame merge

```
df1.merge(df2, on = ['id'])
```

Final data frame is union of both

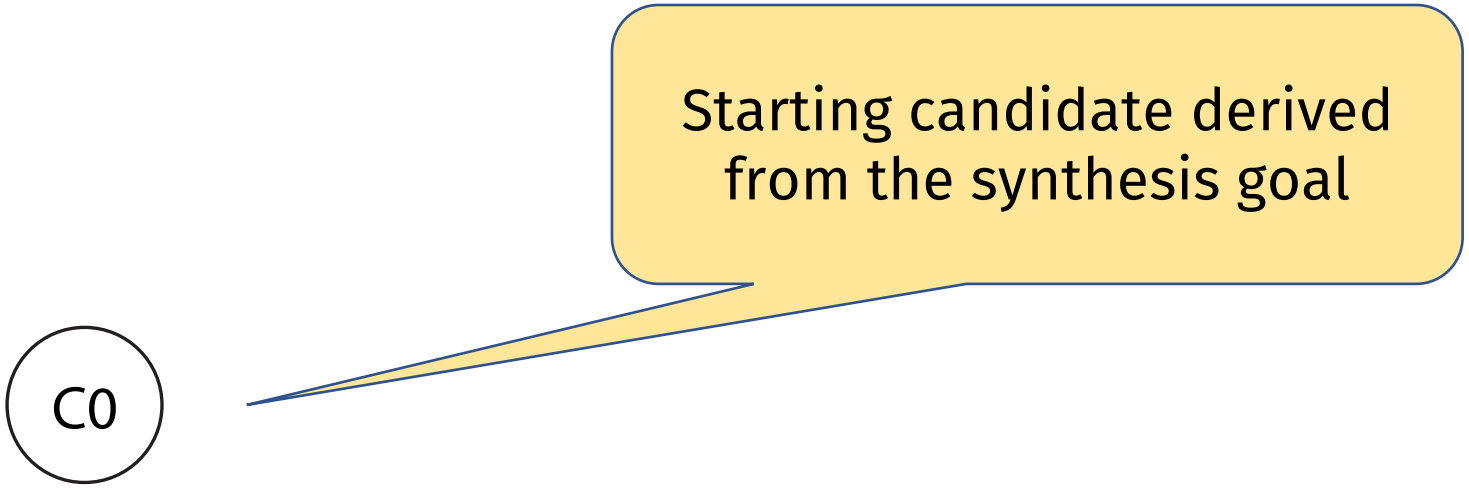
Columns Abstract Interpreter

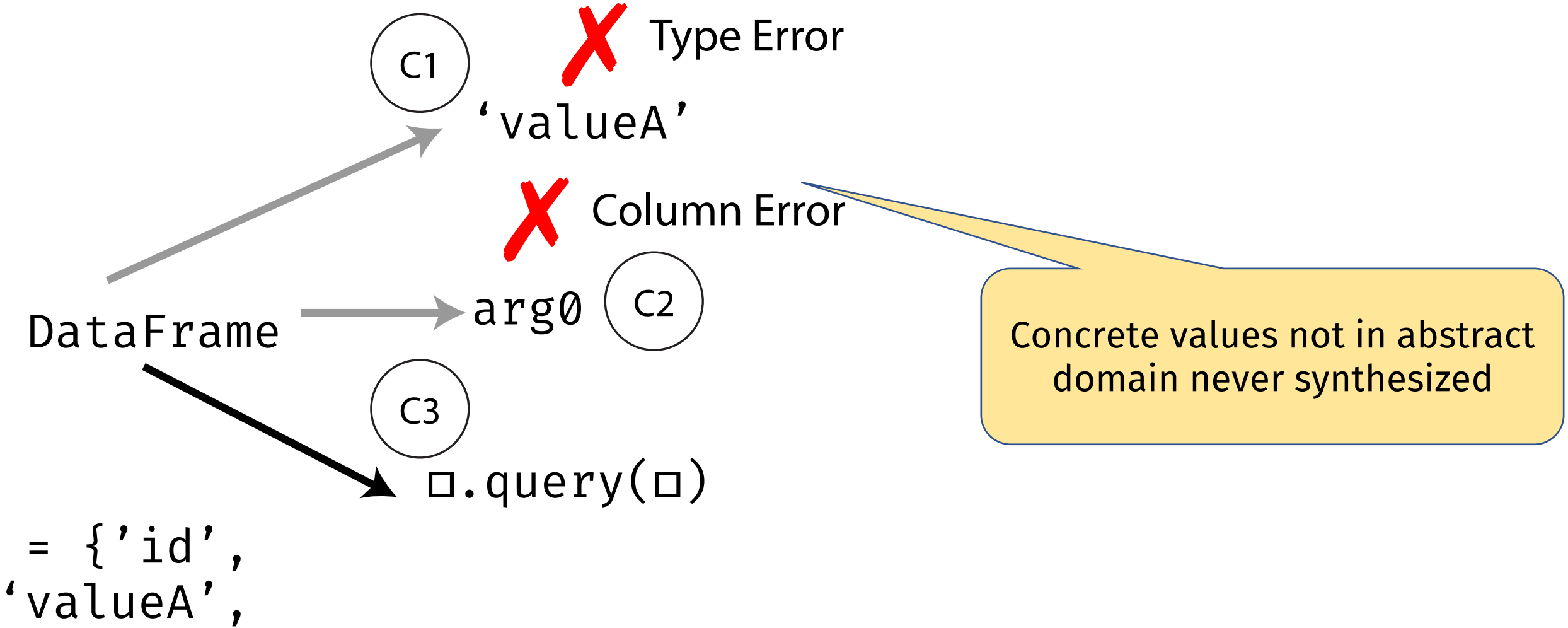
```
class ColNameInterp

  def self.pd_merge(left, right, opt)
    left ∪ right
  end



end
```

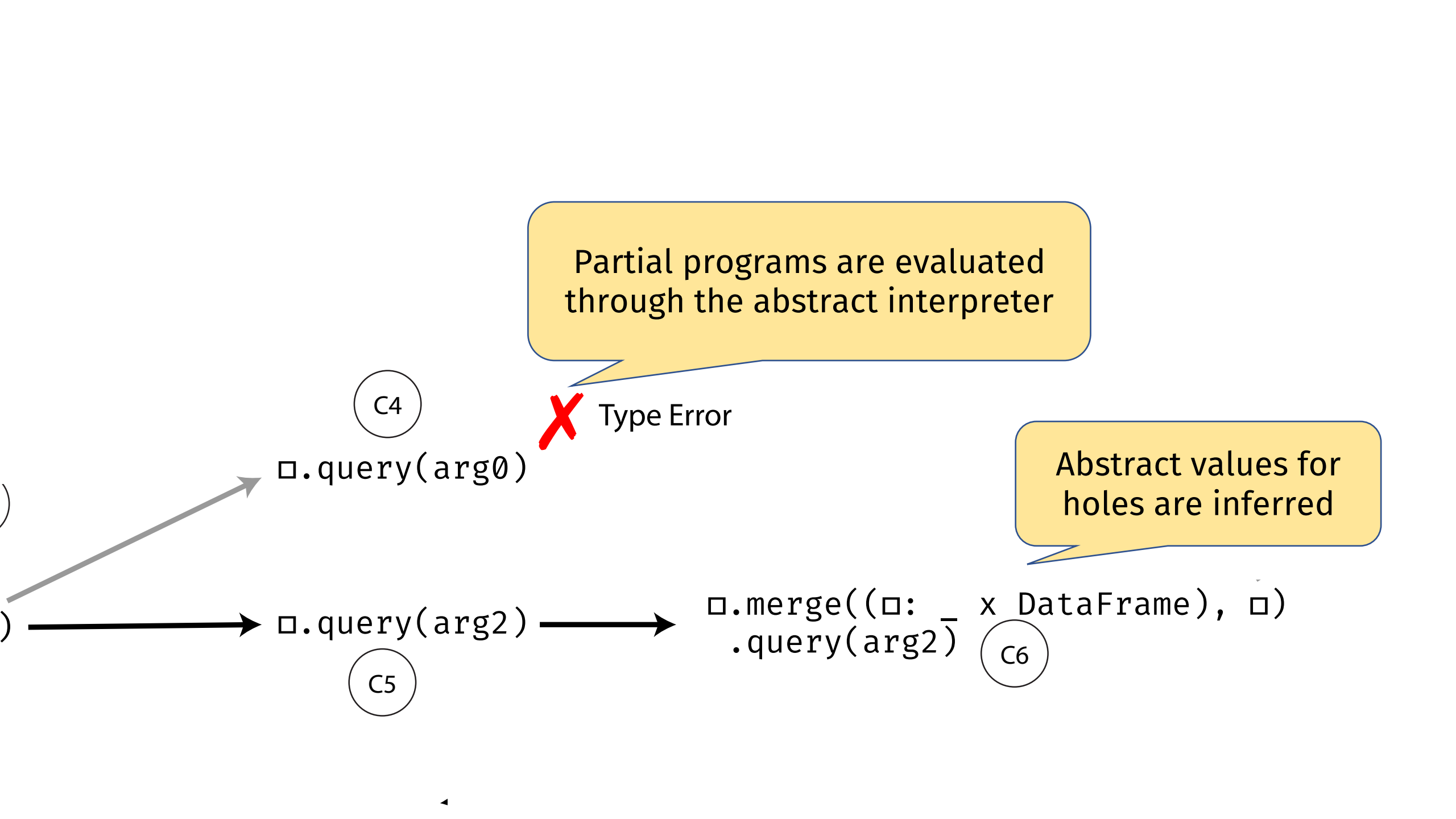
Pandas data frame query

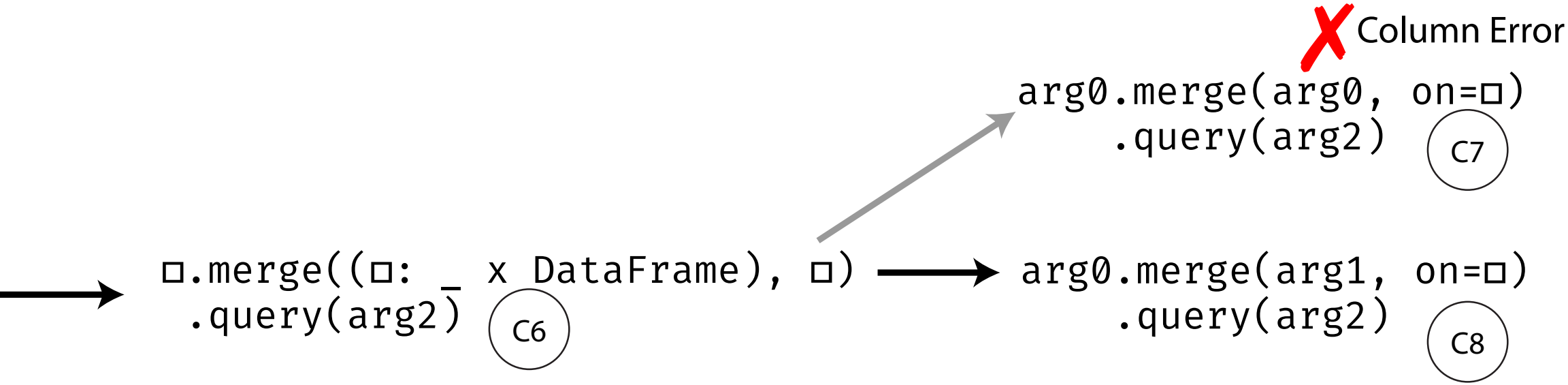
```
df.query('valueA > 10')
```

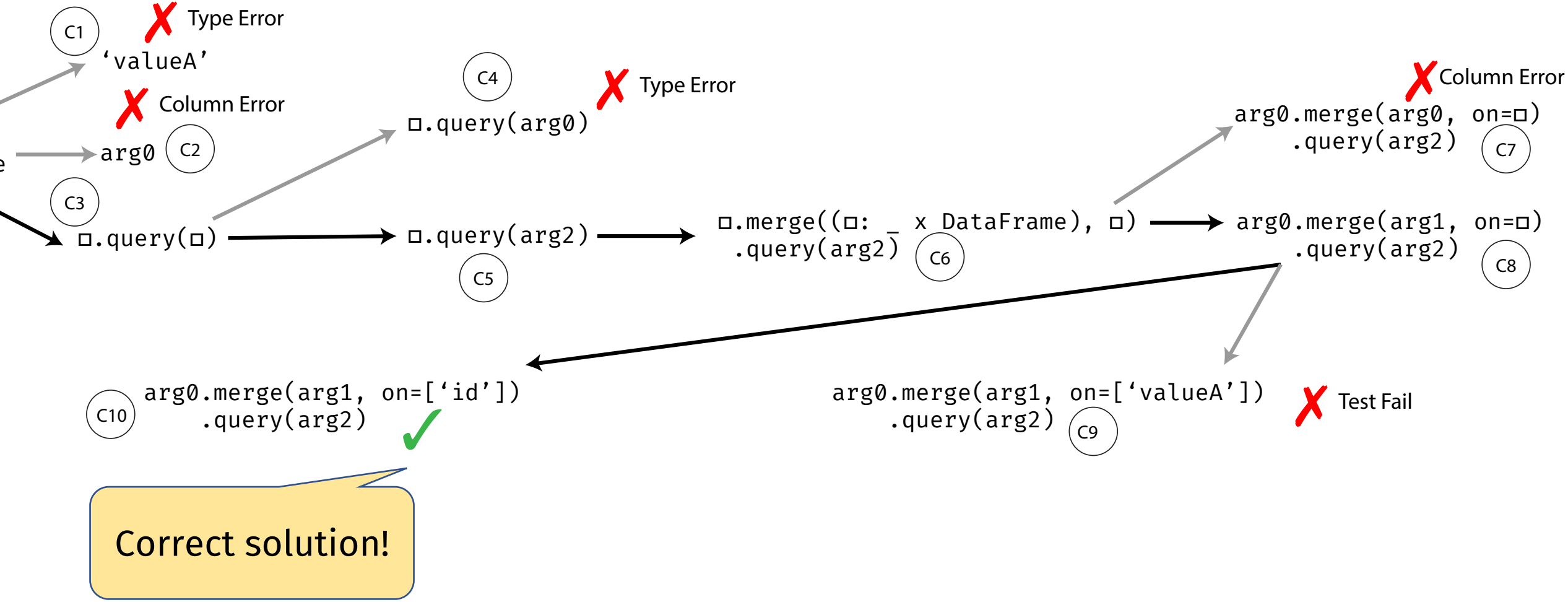
Final data frame has same columns


Starting candidate derived from the synthesis goal
C0
□: Col x DataFrame
Col = {'id',
'valueA',
'valueB'}


C1
Type Error
'valueA'
Column Error
DataFrame
arg0
C2
C3
□.query(□)
= {'id',
'valueA',
Concrete values not in abstract domain never synthesized


Partial programs are evaluated through the abstract interpreter
C4
Type Error
□.query(arg0)
Abstract values for holes are inferred
□.query(arg2)
□.merge((□: _ x DataFrame), □)
.query(arg2)
C5
C6


Column Error
arg0.merge(arg0, on=□)
.query(arg2)
C7
□.merge((□: _ x DataFrame), □)
.query(arg2)
C6
arg0.merge(arg1, on=□)
.query(arg2)
C8

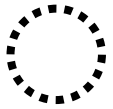

C1
Type Error
'valueA'
Column Error
arg0
C2
C3
□.query(□)
C4
Type Error
□.query(arg0)
□.query(arg2)
C5
□.merge((□: _ x DataFrame), □)
.query(arg2)
C6
Column Error
arg0.merge(arg0, on=□)
.query(arg2)
C7
arg0.merge(arg1, on=□)
.query(arg2)
C8
arg0.merge(arg1, on=['valueA'])
.query(arg2)
C9
Test Fail
C10
arg0.merge(arg1, on=['id'])
.query(arg2)
Correct solution!

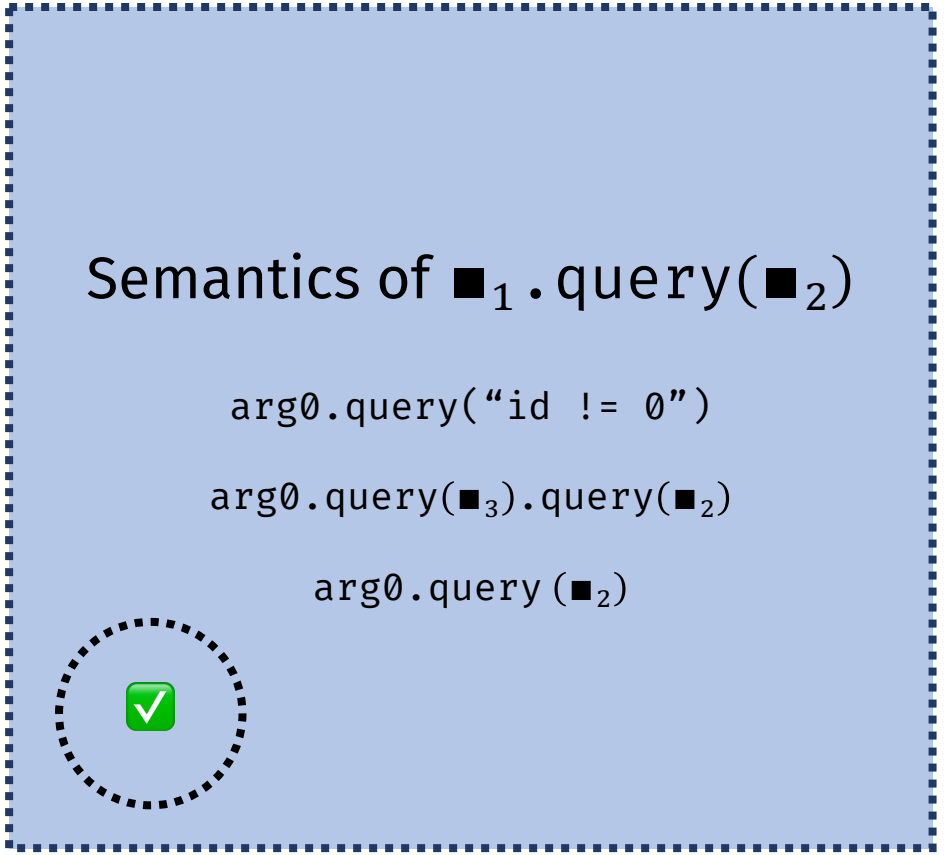
Searching for Programs

Synthesize a term:

$\blacksquare_1$`.query(`$\blacksquare_2$`)`

such that it satisfies a synthesis goal



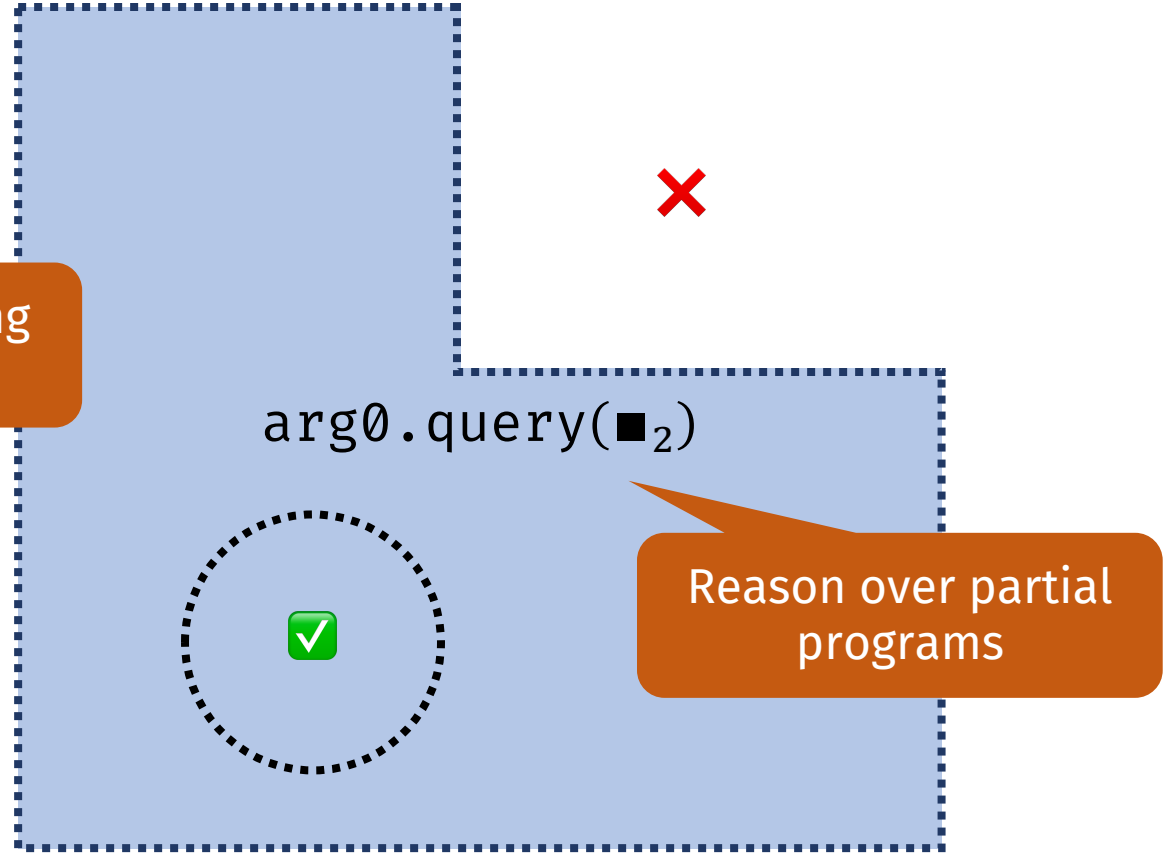


Searching for Programs

Synthesize a term:

`arg0.query(■₂)`

= Synthesis goal



Inferring abstract values

Finite abstract domains:

Types: `Int, Str, DataFrame`

Infinite abstract dom

Solver-aided:

String Length: Linear integer arithmetic

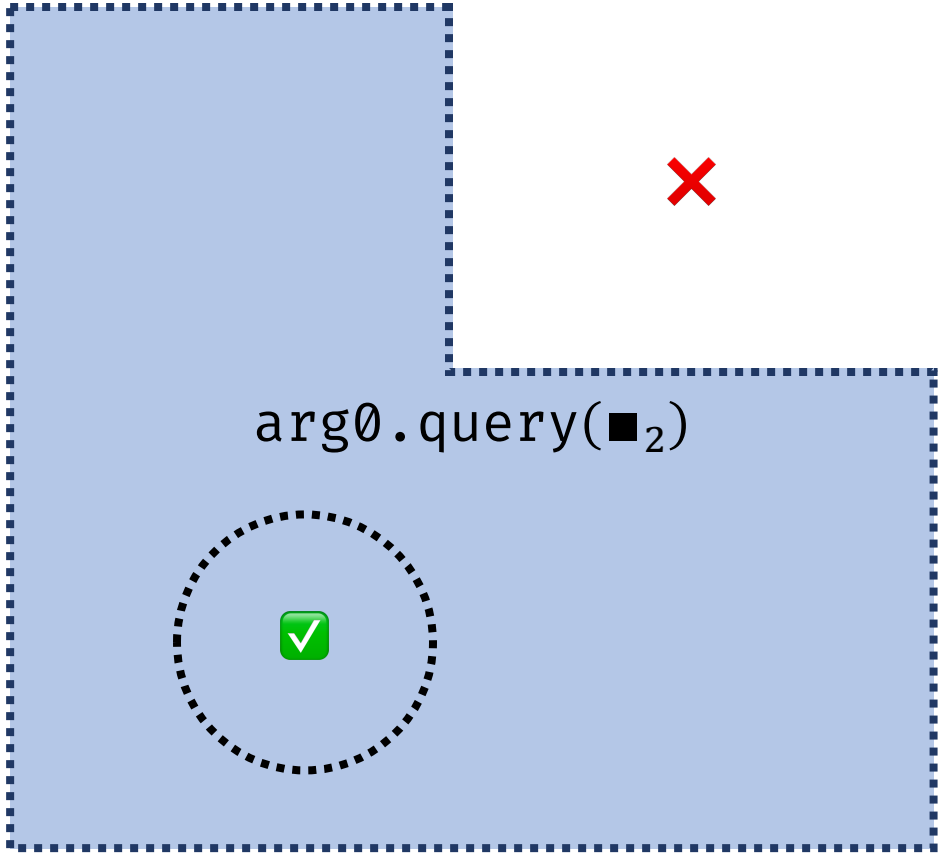
Other:

Data frame columns

Enumerate through valid abstract values

Keep 1 hole symbolic and solve for it

Fall back to term enumeration



Absynthe: Abstract Interpretation-Guided Synthesis

- Abstract domains are good at pruning search space
- Framework uses abstract interpreters as a parameter to guide search
- Abstractions for holes are inferred from abstract semantics
- Solves AutoPandas with simple abstract semantics without GPUs

https://github.com/ngsankha/absynthe