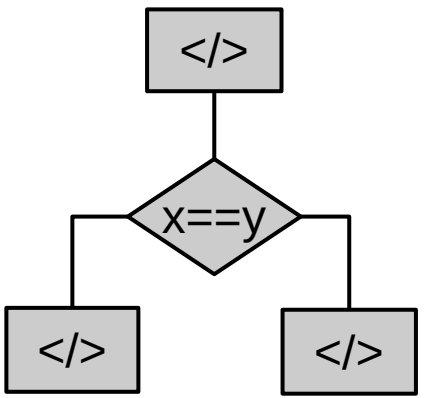
Verification Algorithms for Automated Separation Logic Verifiers

Marco Eilers
Malte Schwerhoff
Peter Müller

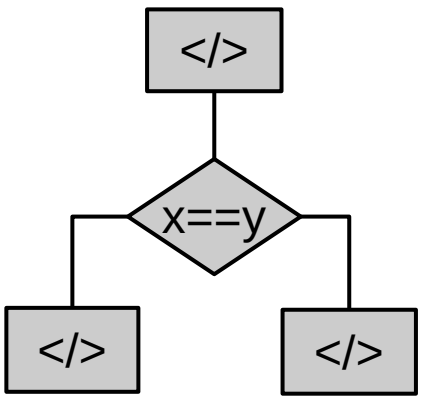
ETH zürich

Automating Separation Logic



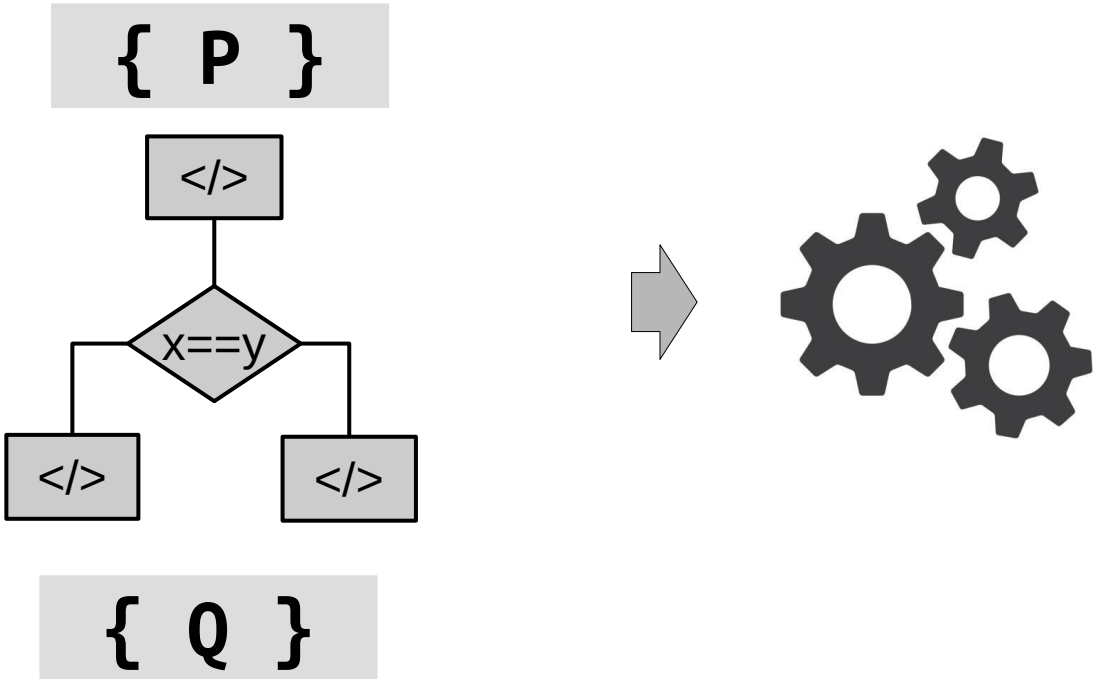
Automating Separation Logic

{ P }

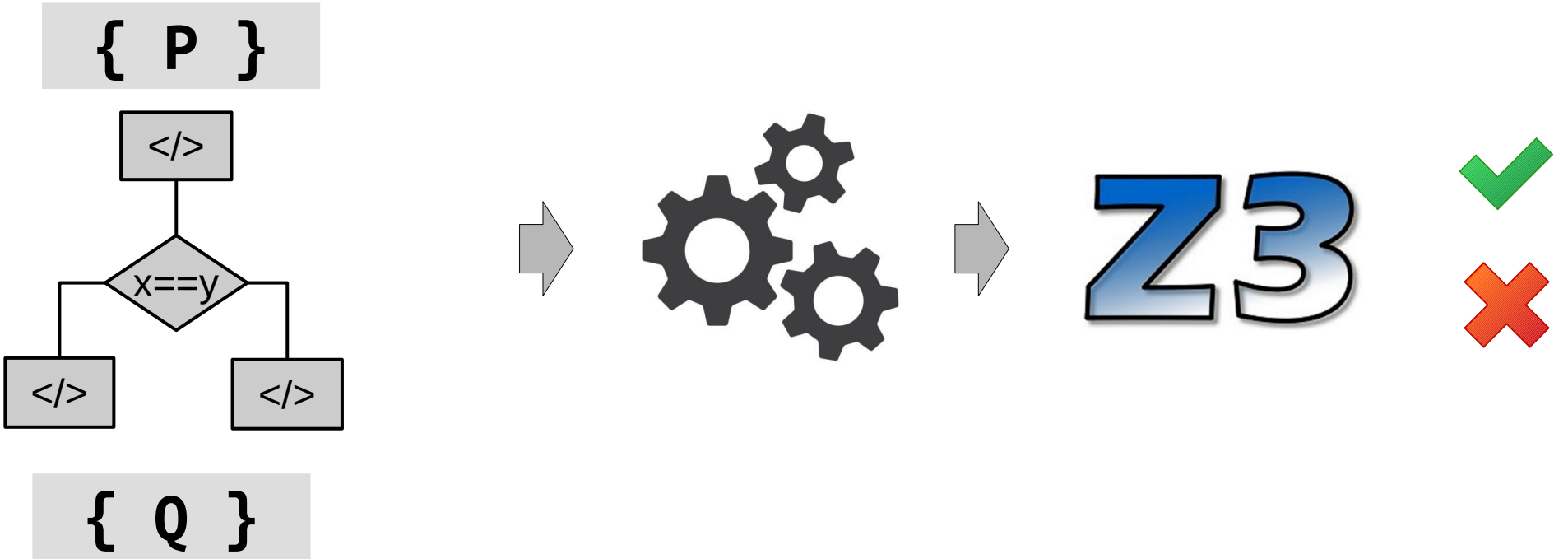


{ Q }

Automating Separation Logic

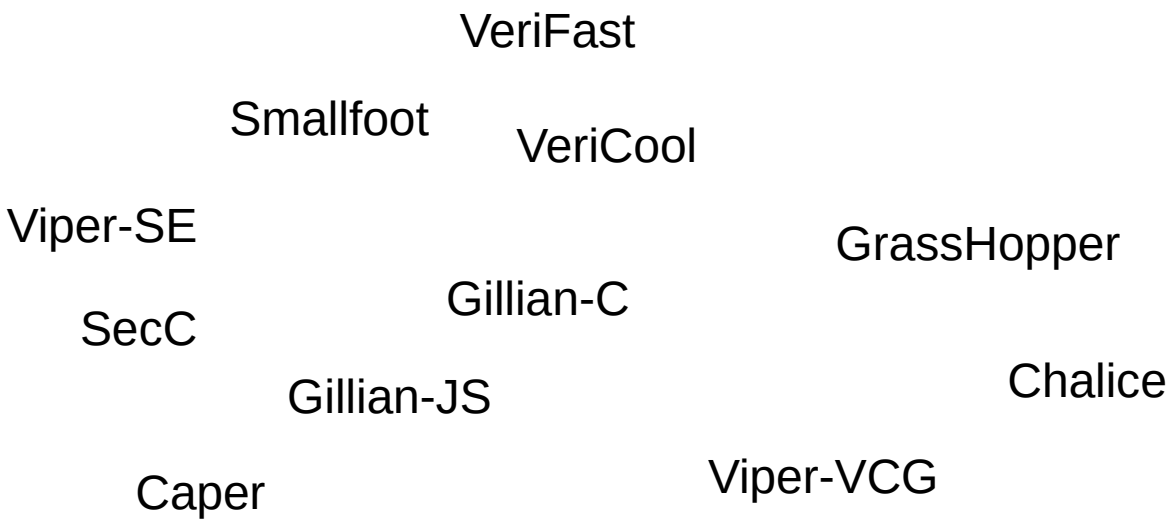


Automating Separation Logic

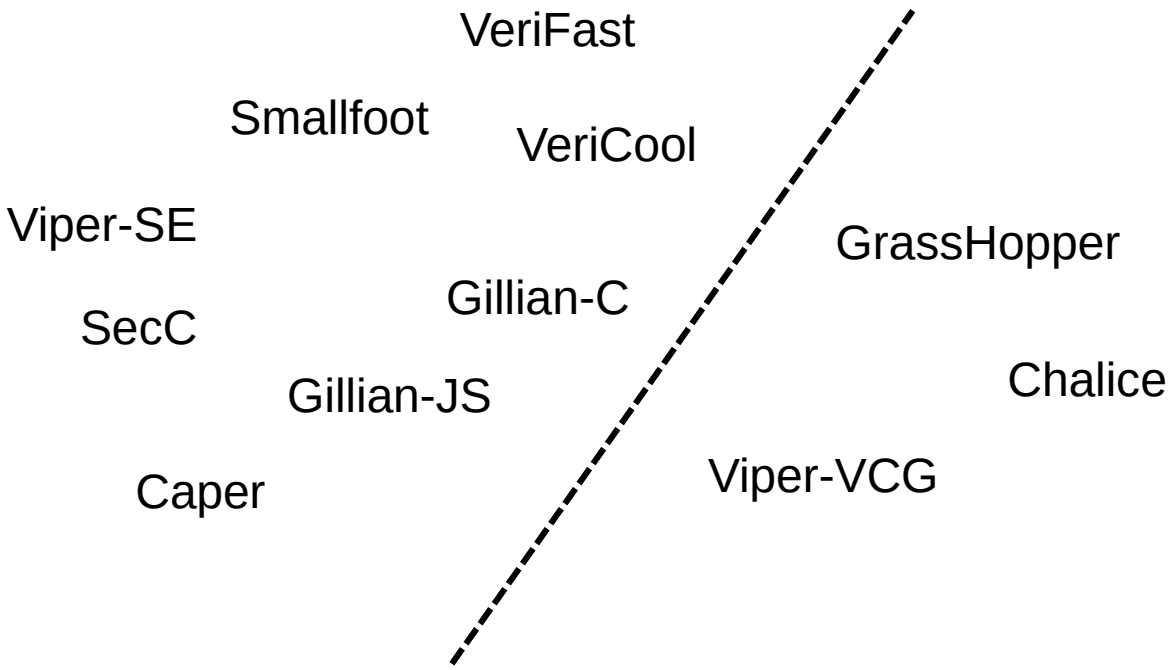


What verification algorithms exist
and
which is the best?

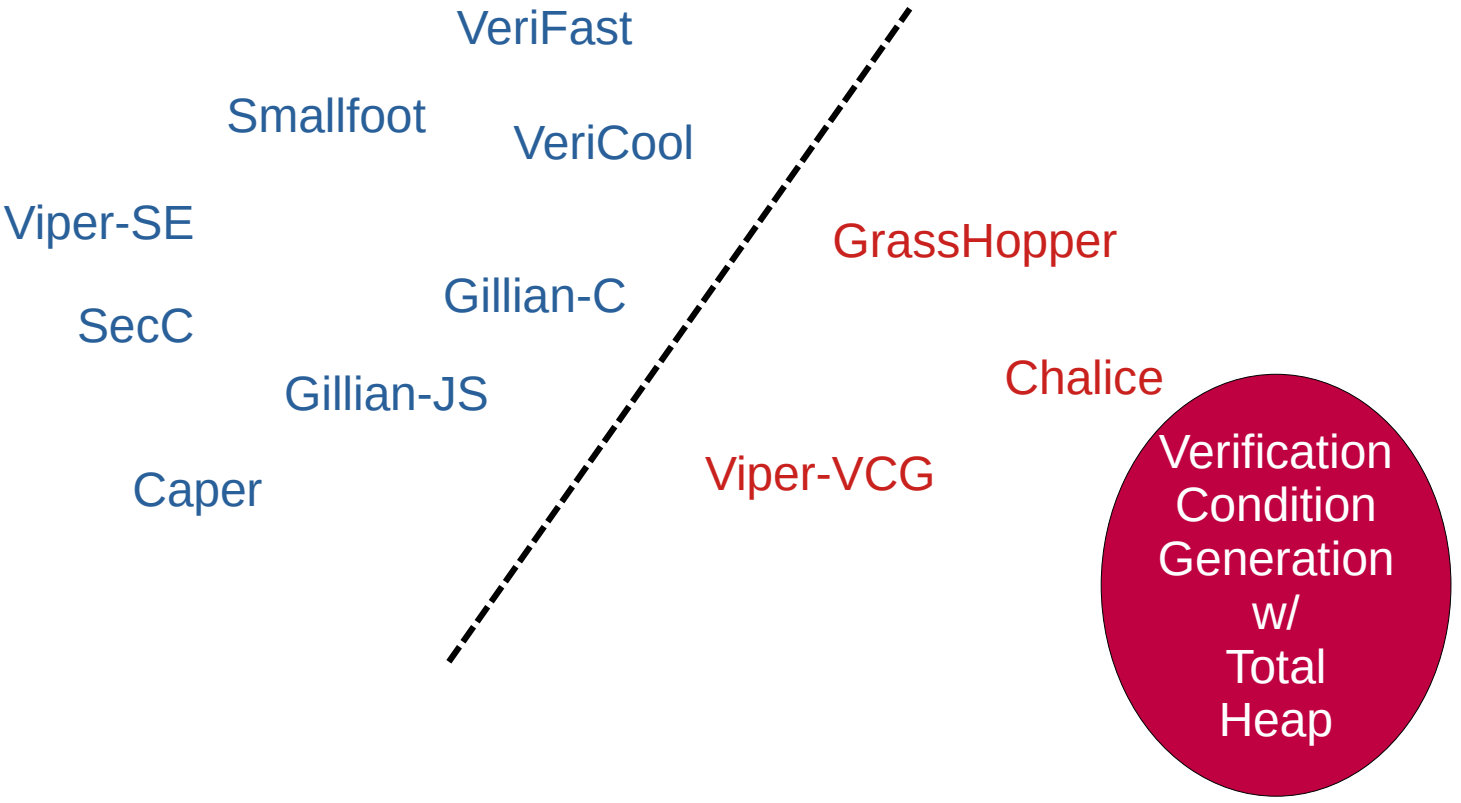
Automated Separation Logic Verification Tools



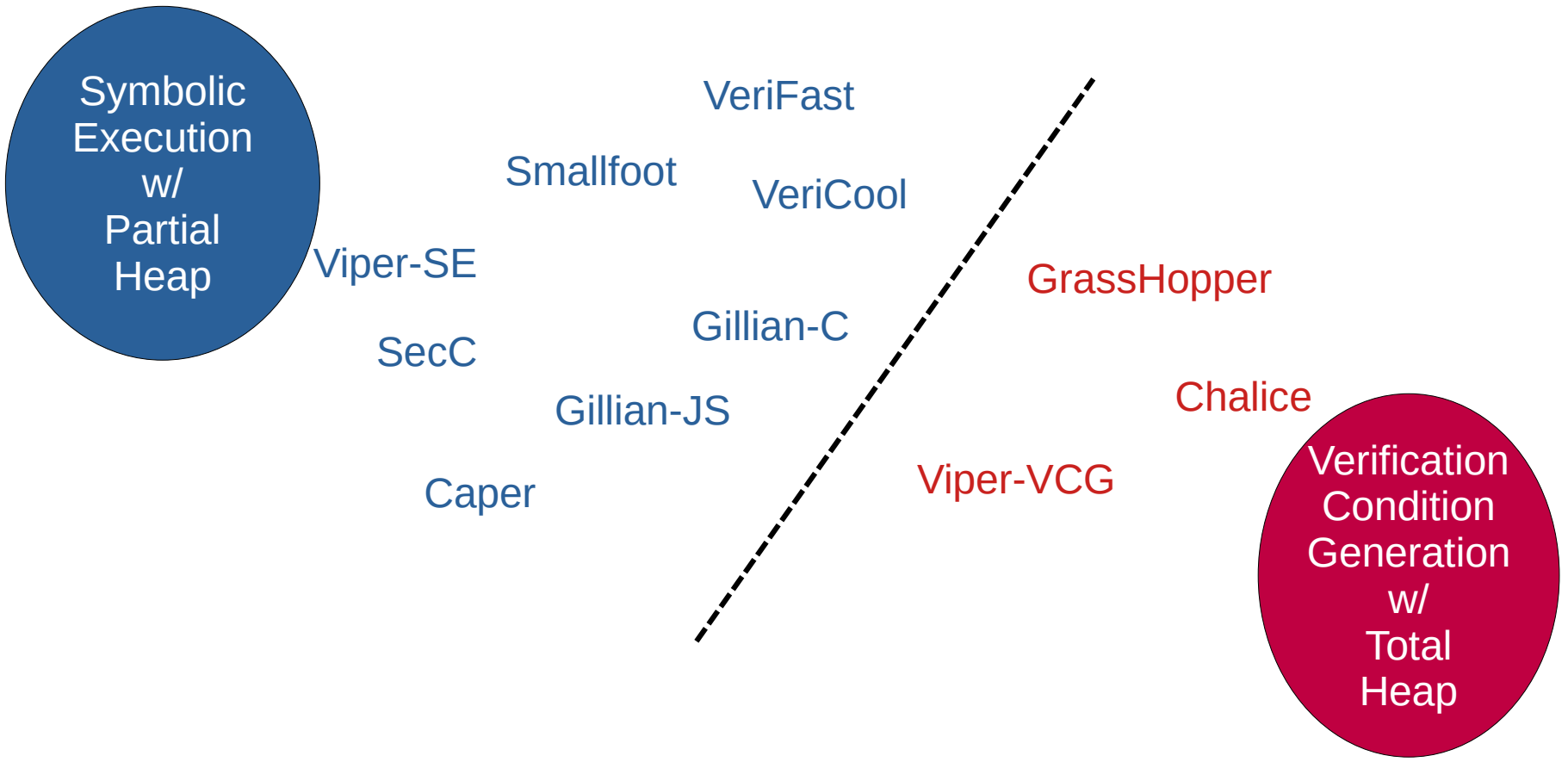
Automated Separation Logic Verification Tools



Automated Separation Logic Verification Tools



Automated Separation Logic Verification Tools



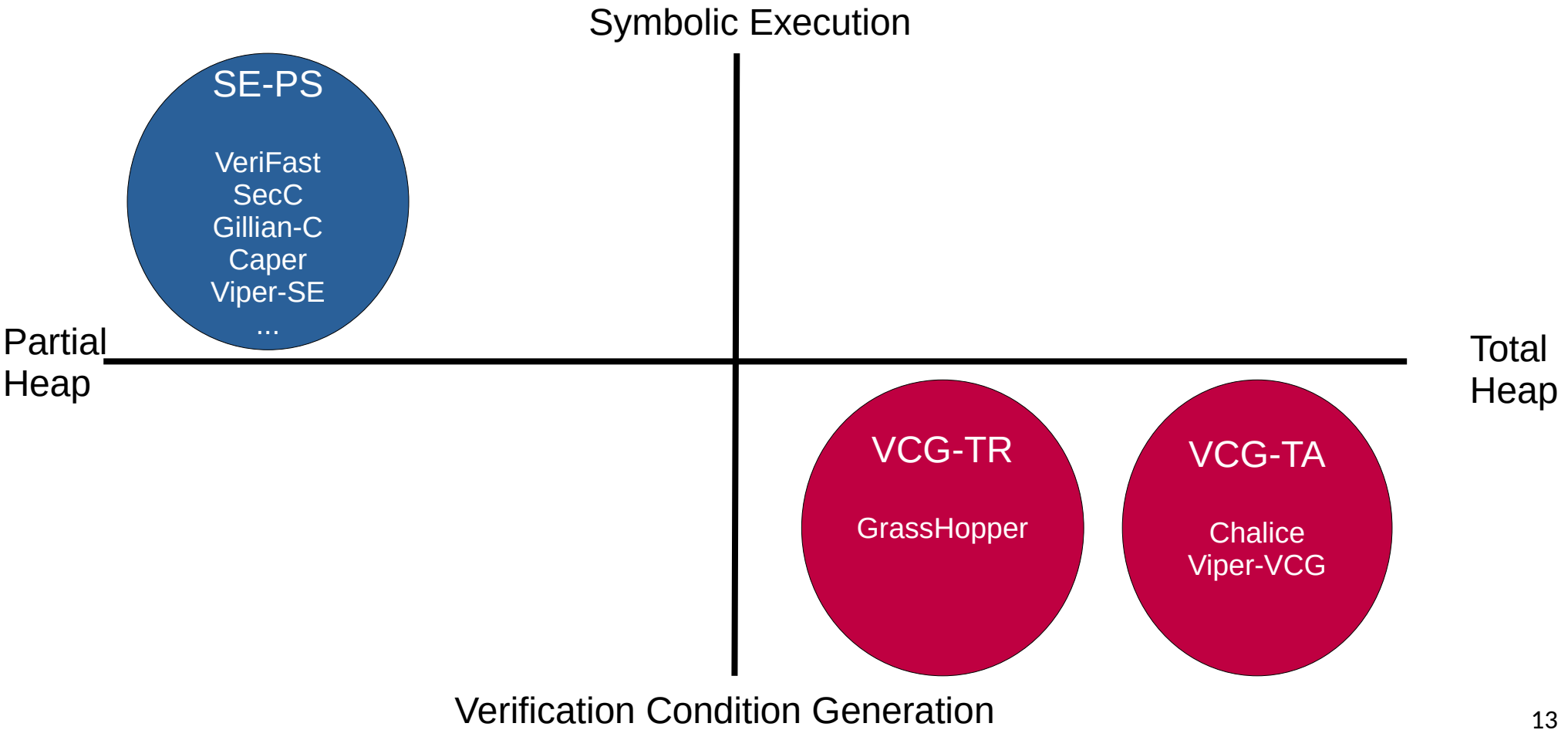
Contributions

- Survey of algorithms used in existing separation logic verifiers

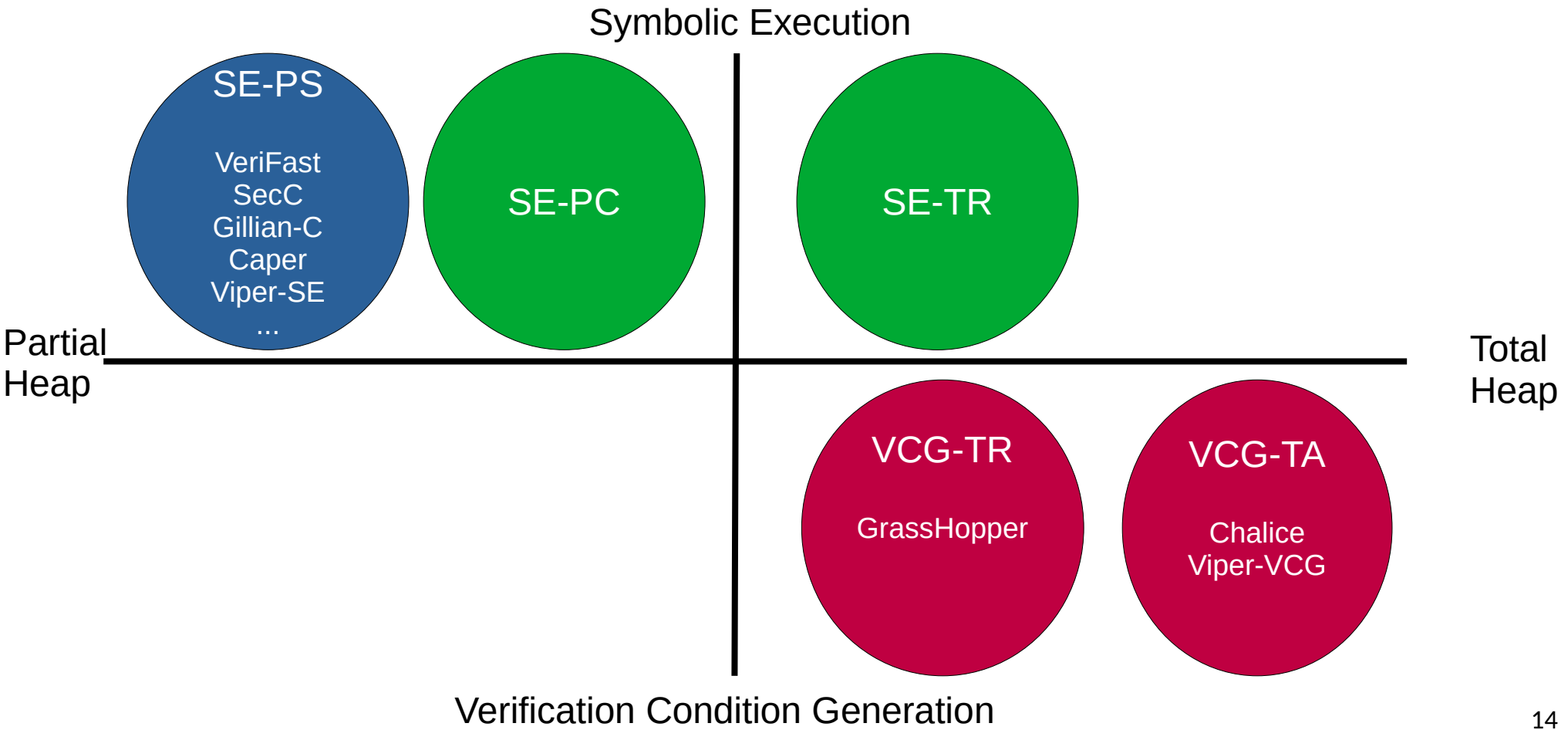
Contributions

- Survey of algorithms used in existing separation logic verifiers
- Propose two new algorithms

Preview: Algorithm Comparison



Preview: Algorithm Comparison



Contributions

- Survey of algorithms used in existing separation logic verifiers
- Propose two new algorithms

- Implement all five algorithms in Viper
  - Two already existed

Contributions

- Survey of algorithms used in existing separation logic verifiers
- Propose two new algorithms

- Implement all five algorithms in Viper
  - Two already existed

- Systematic empirical evaluation of completeness and performance

Contributions

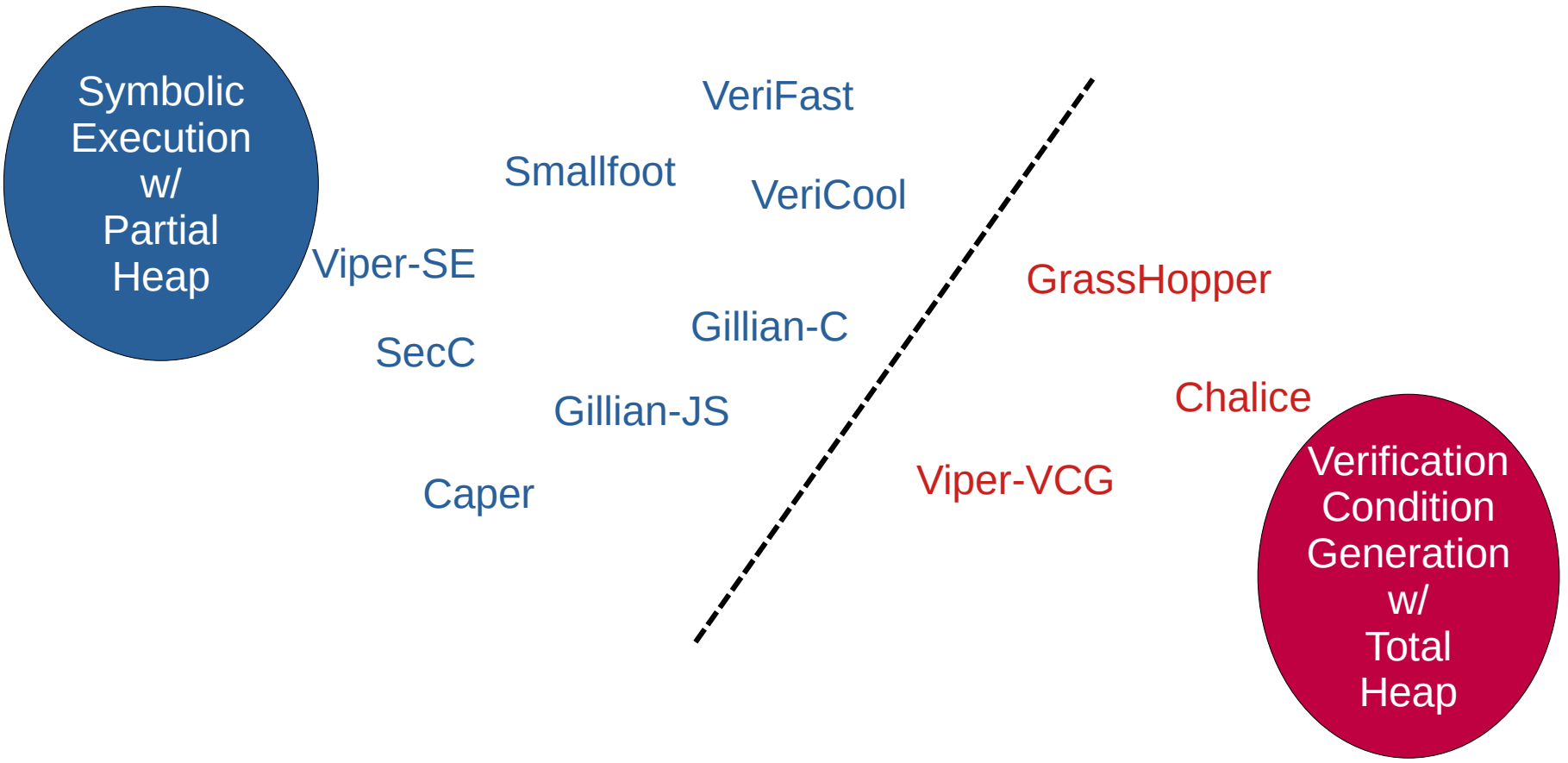
- Survey of algorithms used in existing separation logic verifiers
- Propose two new algorithms

- Implement all five algorithms in Viper
  - Two already existed

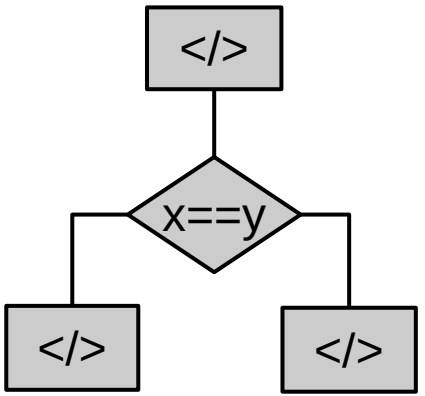
- Systematic empirical evaluation of completeness and performance

- Comparison of different portfolios

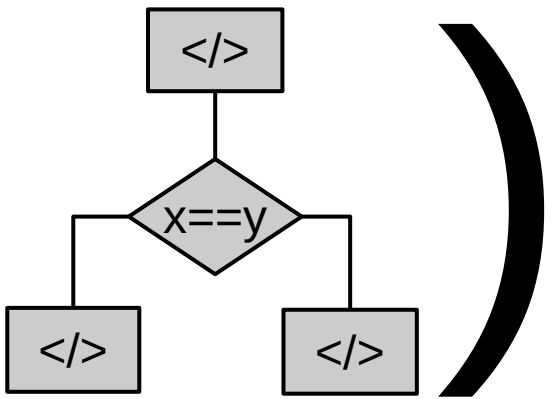
Automated Separation Logic Verification Tools



Verification Condition Generation (VCG)

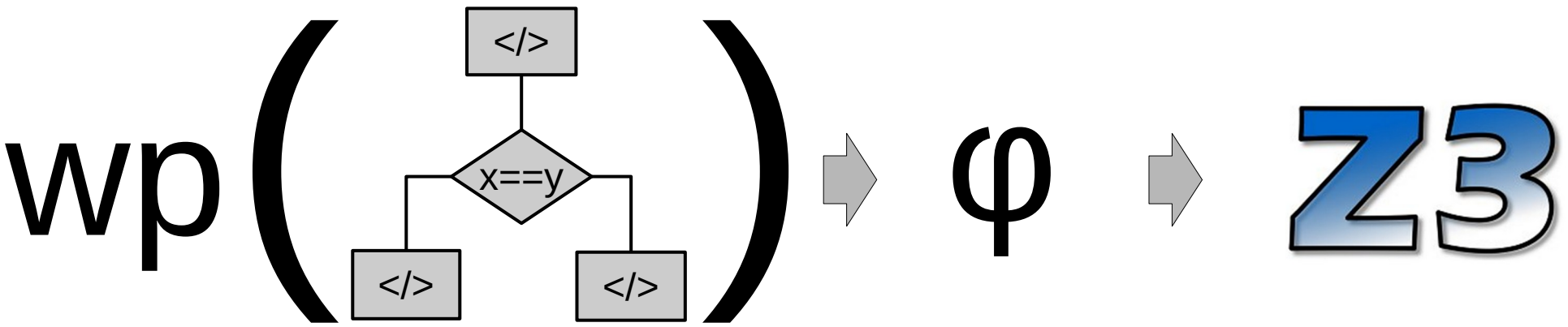


Verification Condition Generation (VCG)

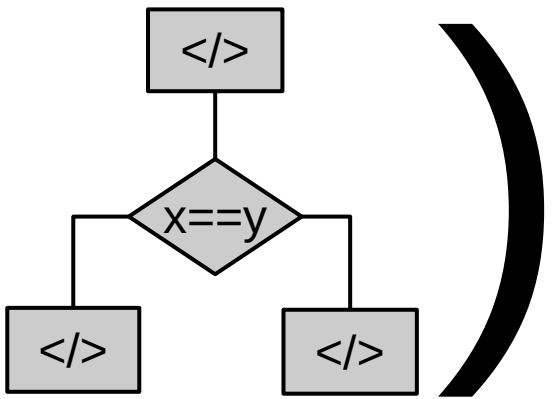
wp(

)

Verification Condition Generation (VCG)

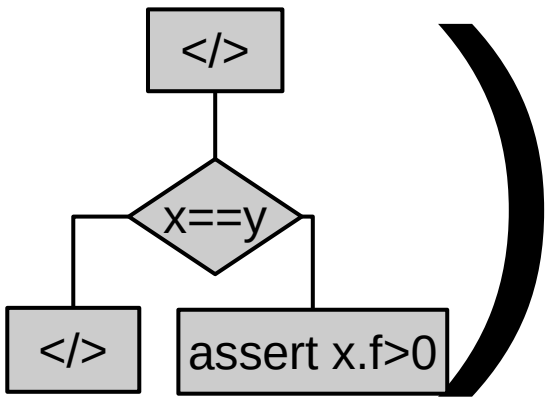


VCG with Total Heaps

wp(

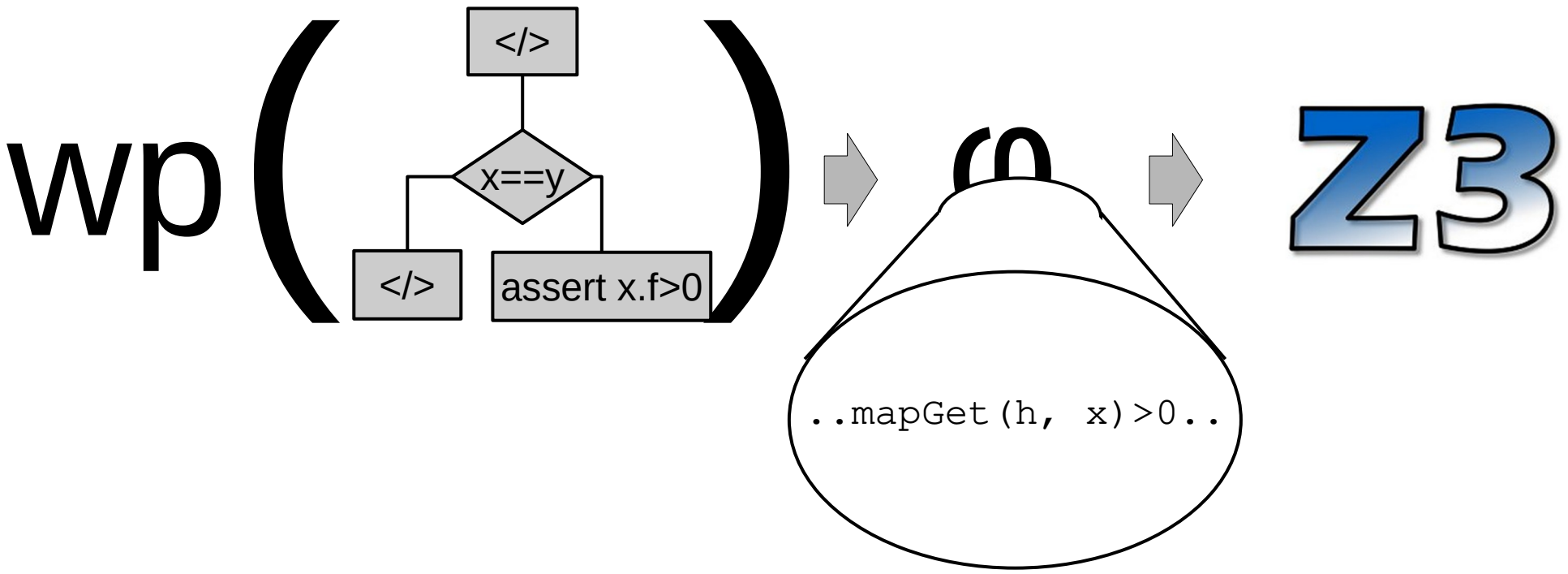
) ⇨ φ ⇨ Z3

VCG with Total Heaps

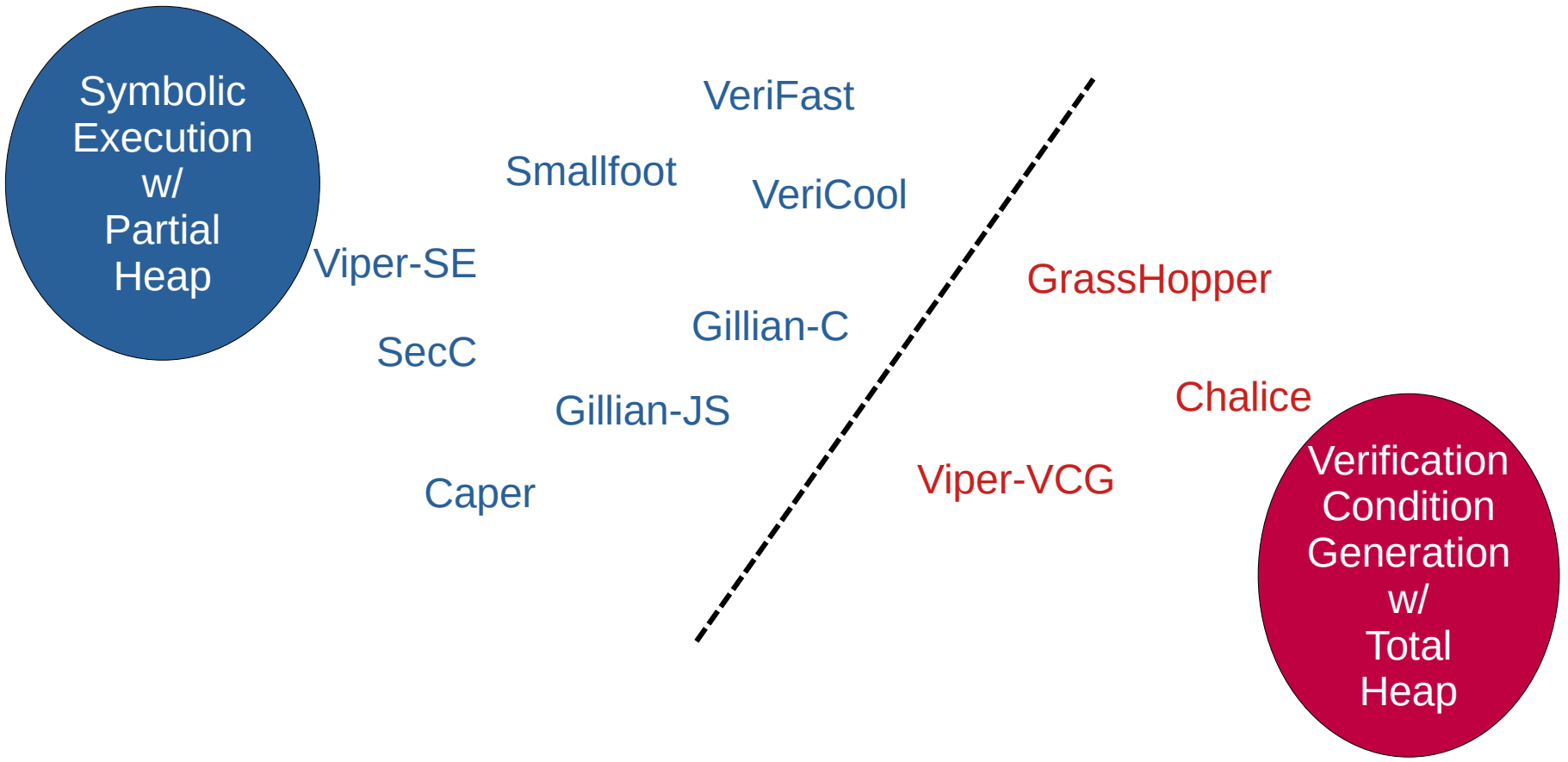
wp(

) ➡ φ ➡ Z3

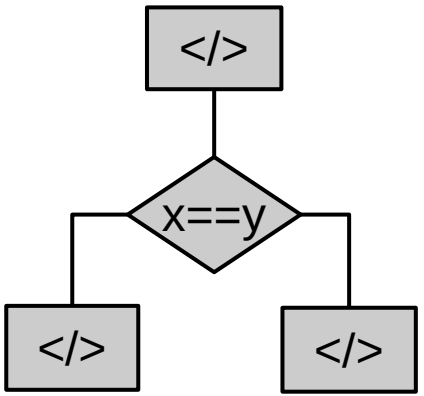
VCG with Total Heaps



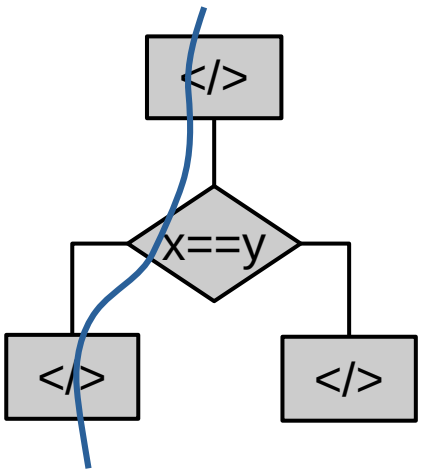
Automated Separation Logic Verification Tools



Symbolic Execution (SE)



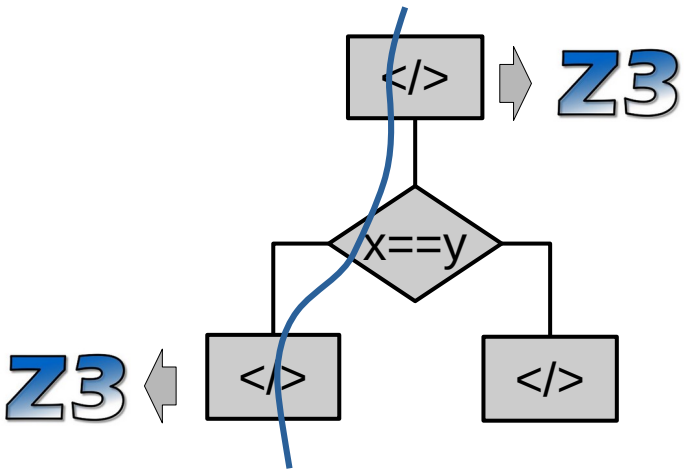
Symbolic Execution (SE)



Store

```
X → x0
y → 1
```

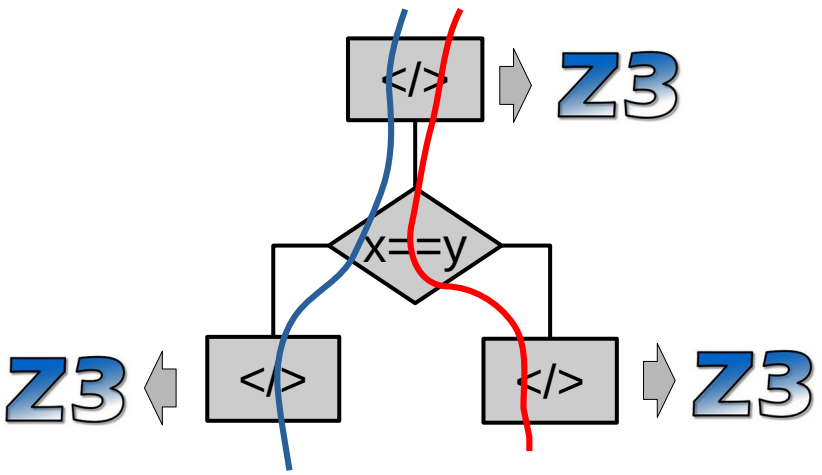
Symbolic Execution (SE)



Store

```
X → x0
y → 1
```

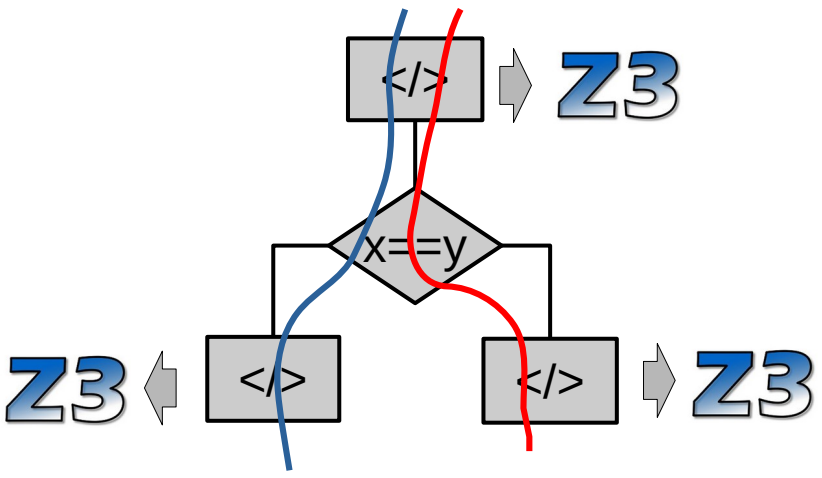
Symbolic Execution (SE)



Store

```
X → x0
y → 1
```

SE with Partial Heaps



Store

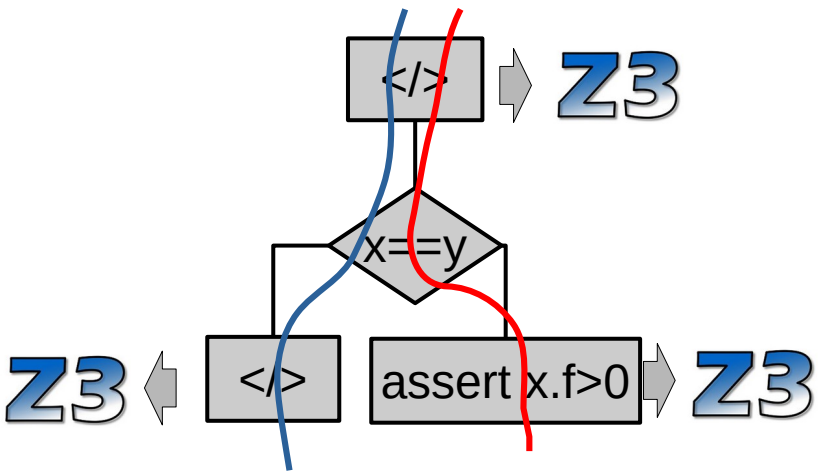
```
x → x0
y → 1
```

Heap

z0.f → v0

x0.f → v1

SE with Partial Heaps



Store

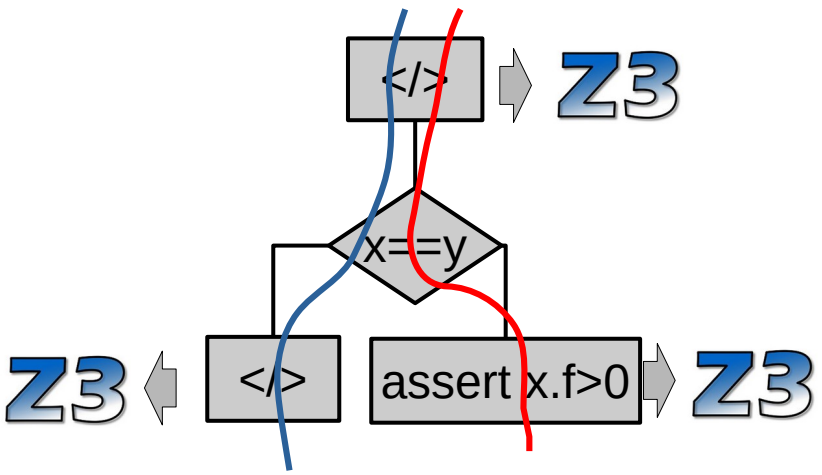
```
x → x0
y → 1
```

Heap

z0.f → v0

x0.f → v1

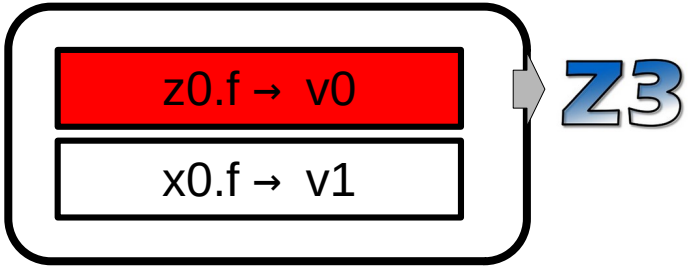
SE with Partial Heaps



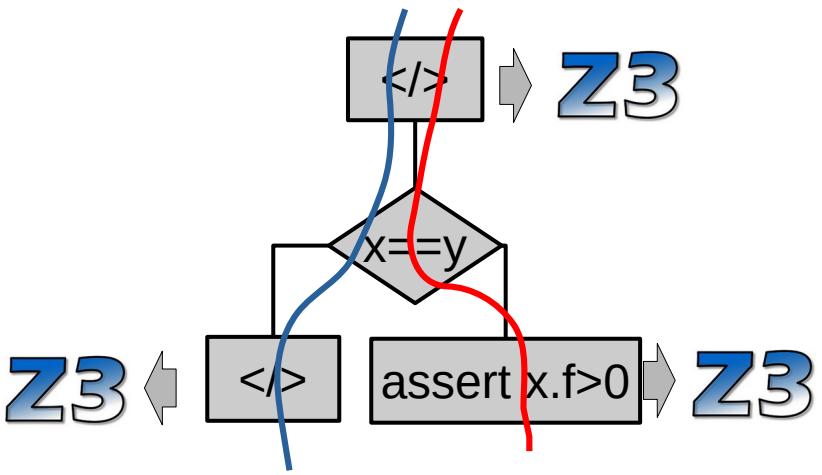
Store

```
x → x0
y → 1
```

Heap



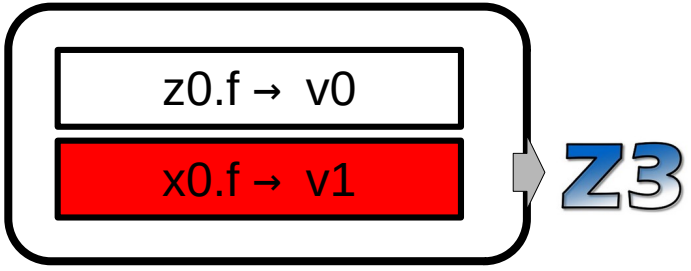
SE with Partial Heaps



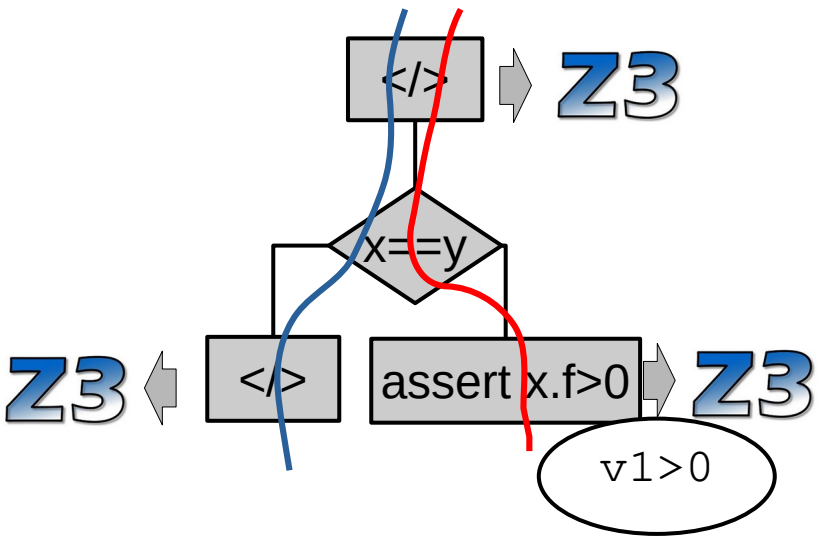
Store

```
x → x0
y → 1
```

Heap



SE with Partial Heaps



Store

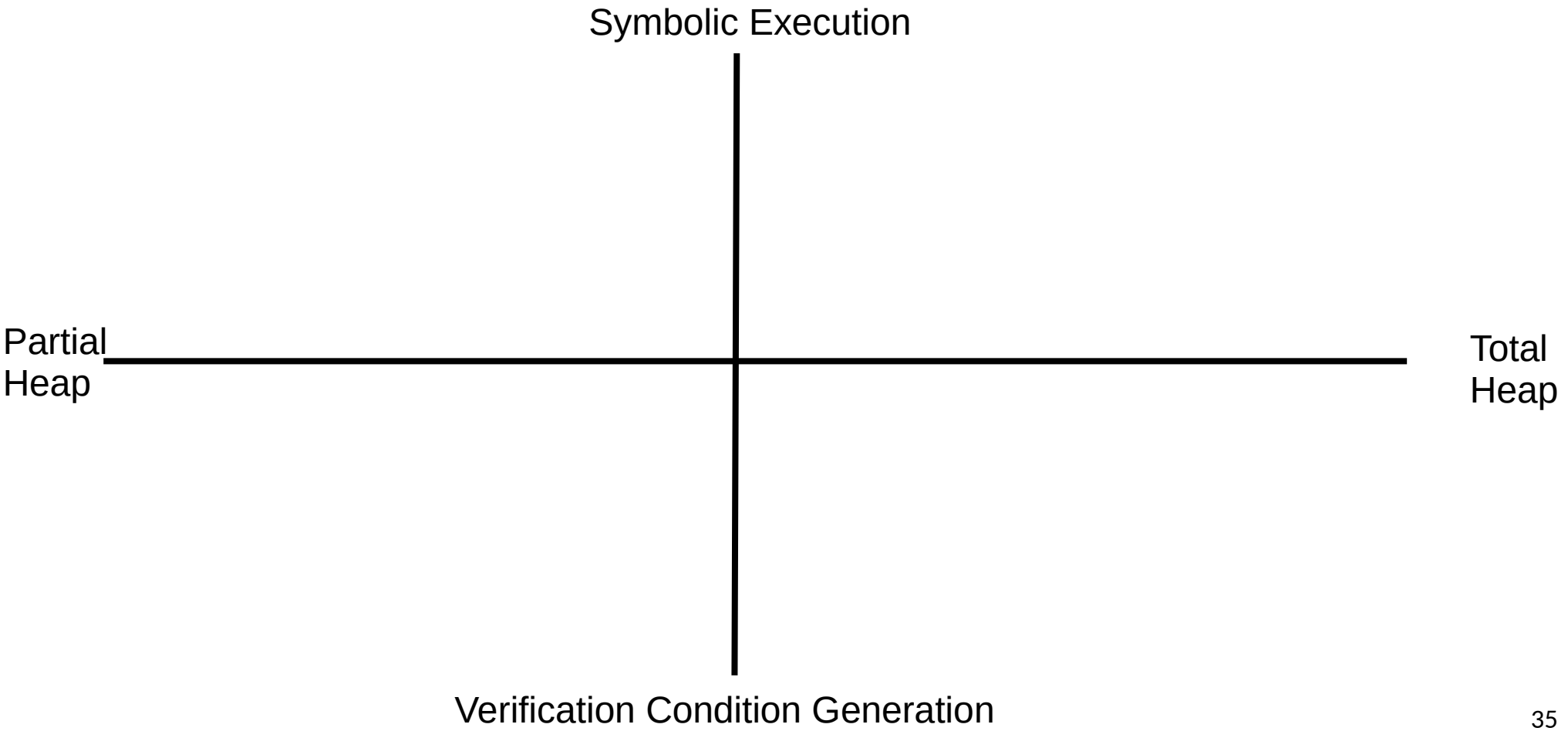
```
x → x0
y → 1
```

Heap

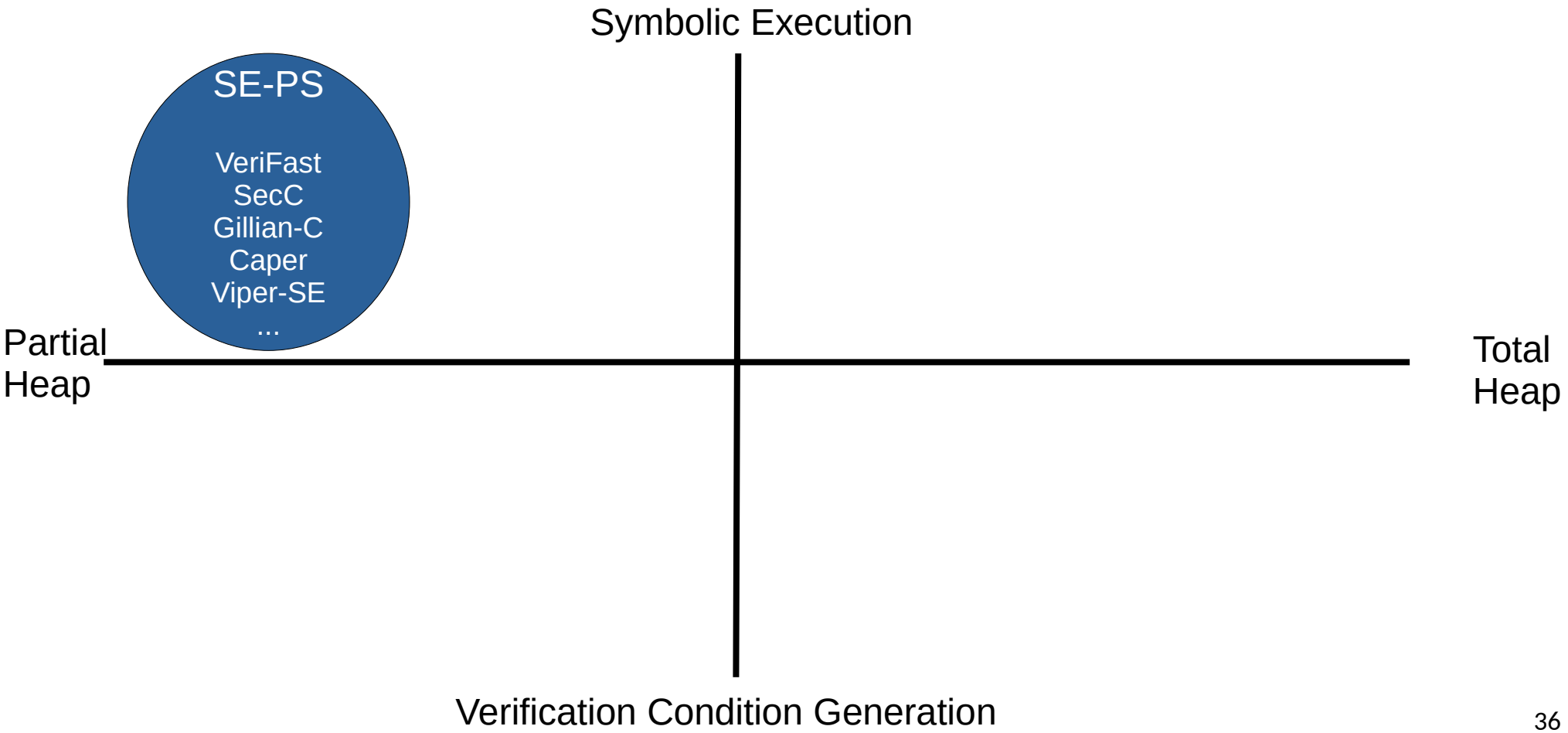
z0.f → v0

x0.f → v1

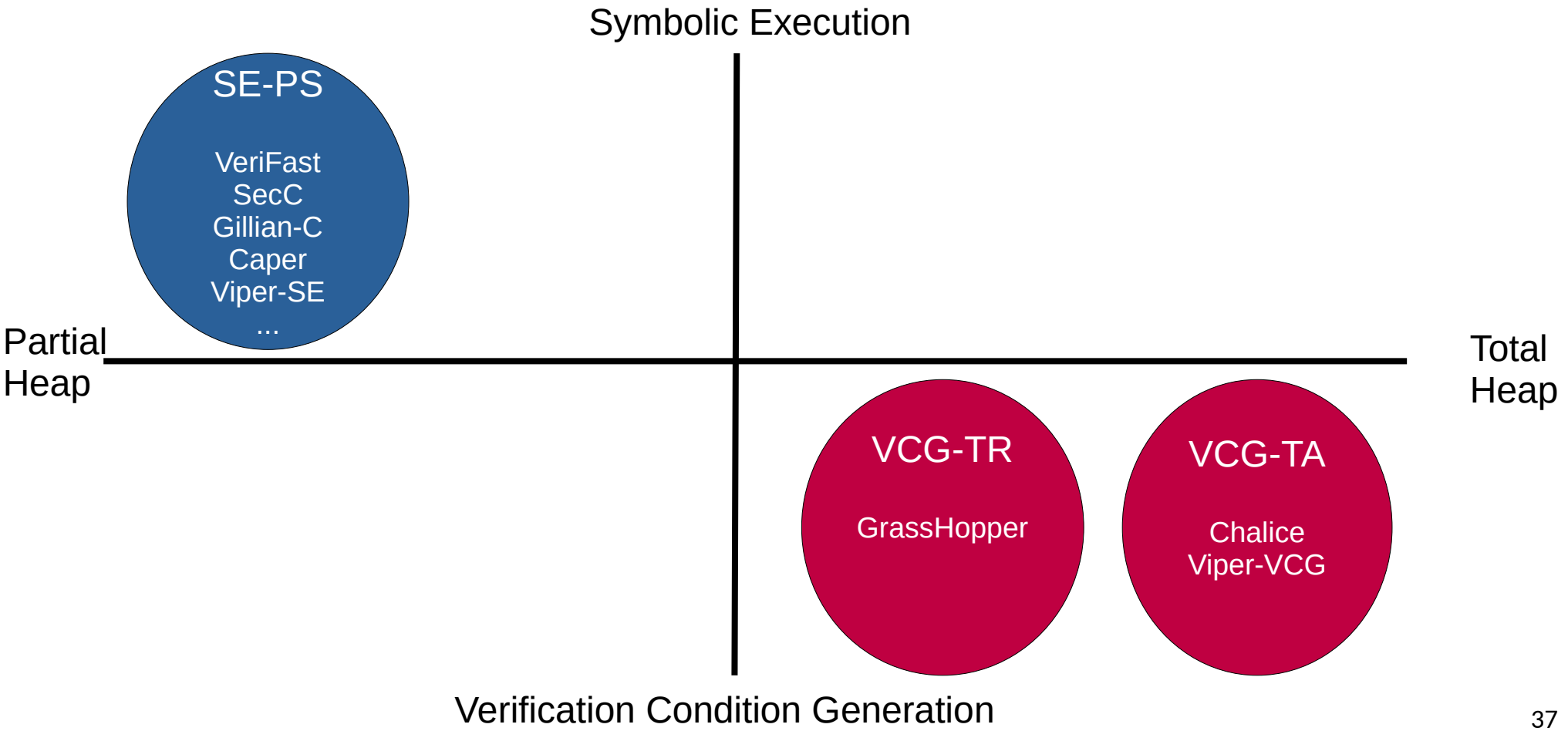
Algorithm Comparison



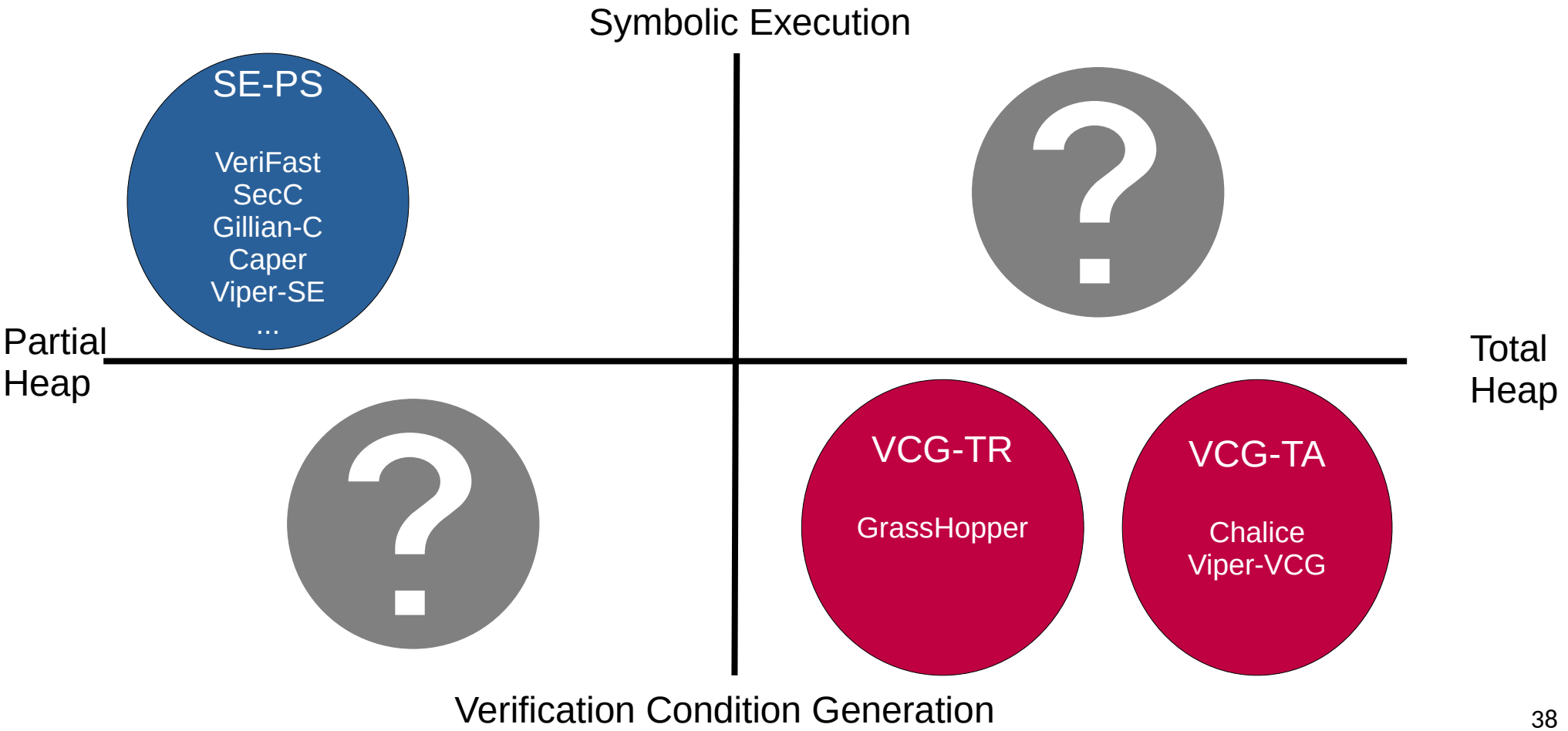
Algorithm Comparison



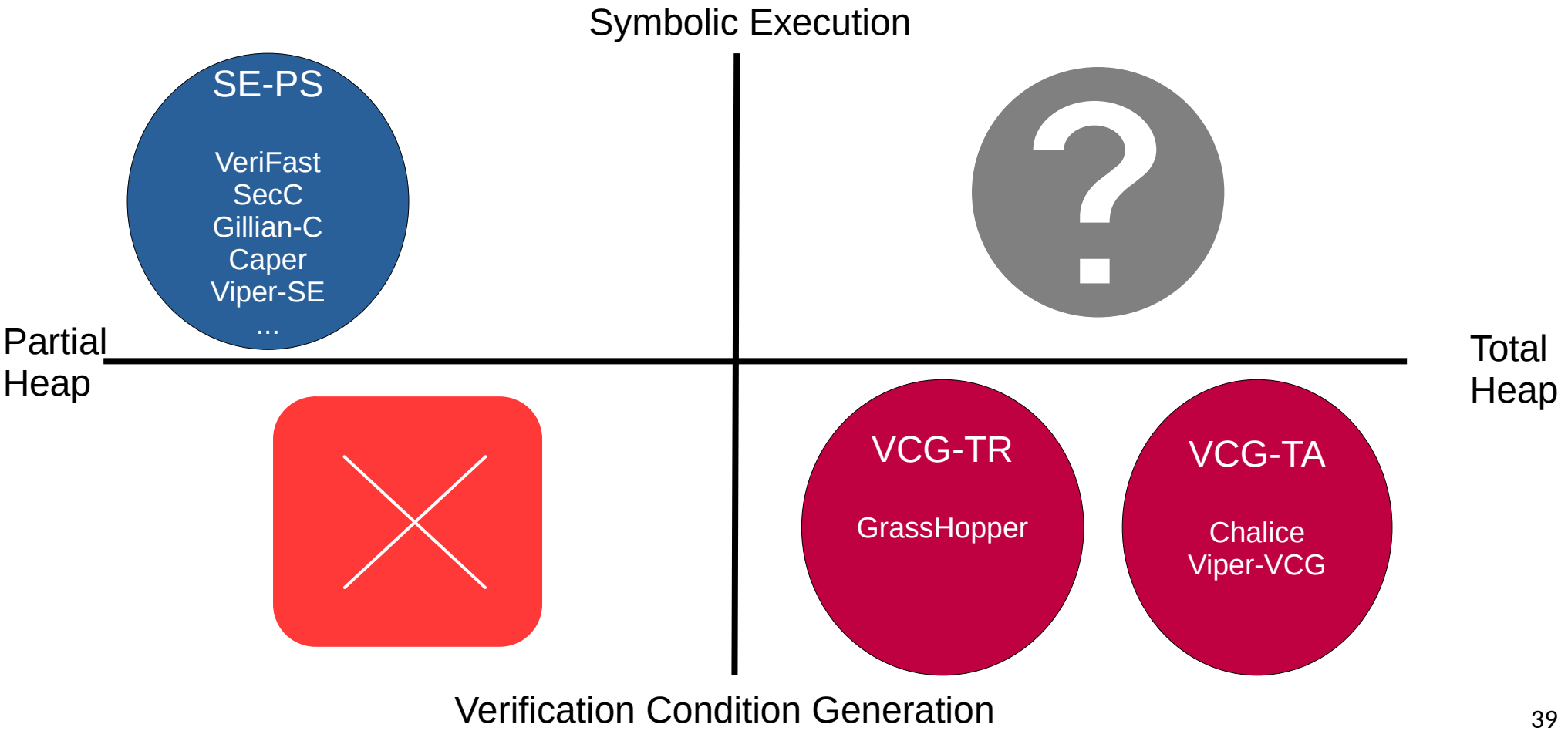
Algorithm Comparison



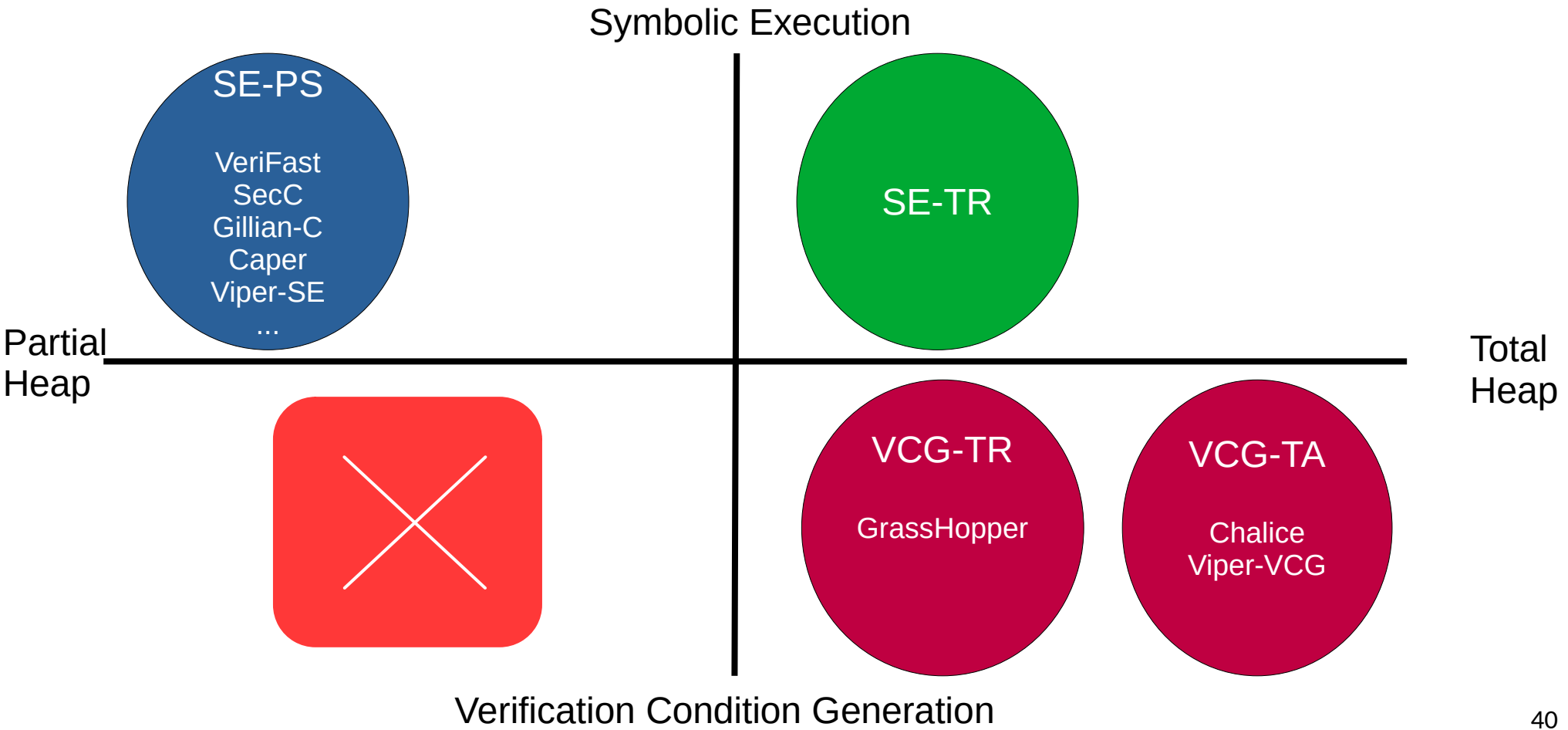
Algorithm Comparison



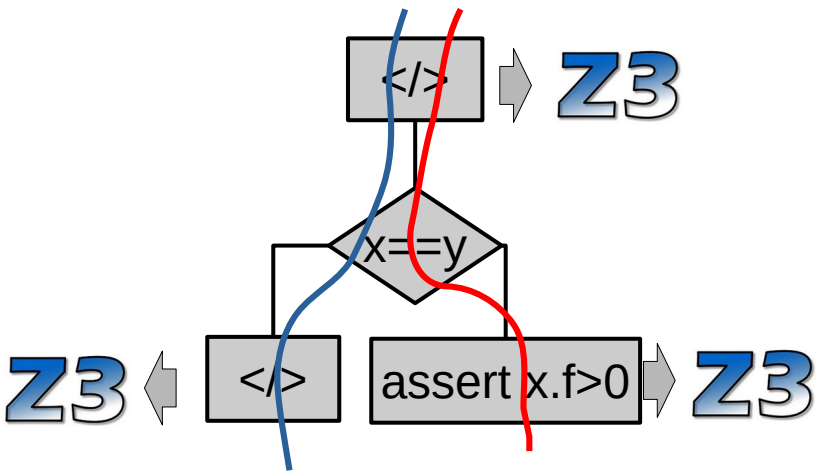
Algorithm Comparison



Algorithm Comparison



Symbolic Execution with Total Heaps

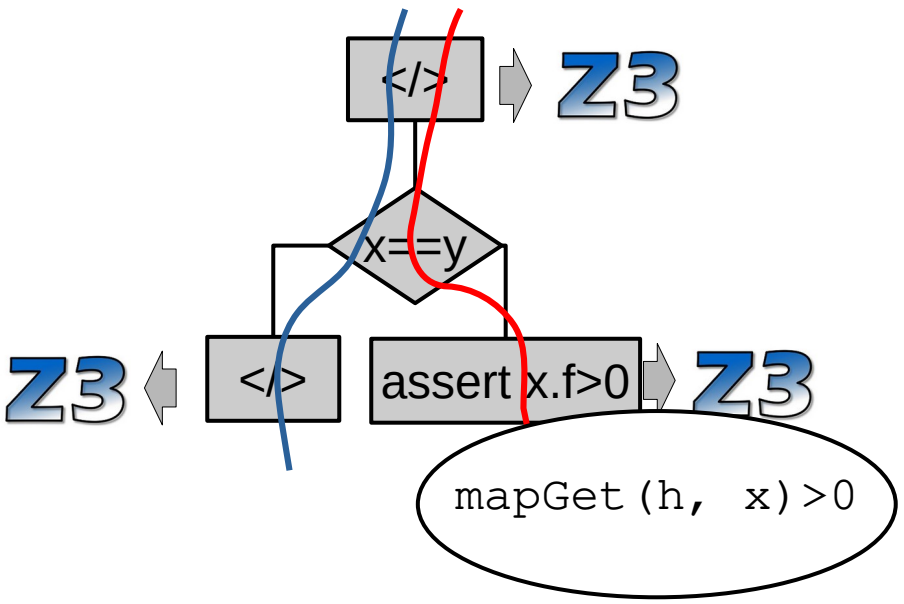


Store

```
x → x0
y → 1
```

Heap: h

Symbolic Execution with Total Heaps

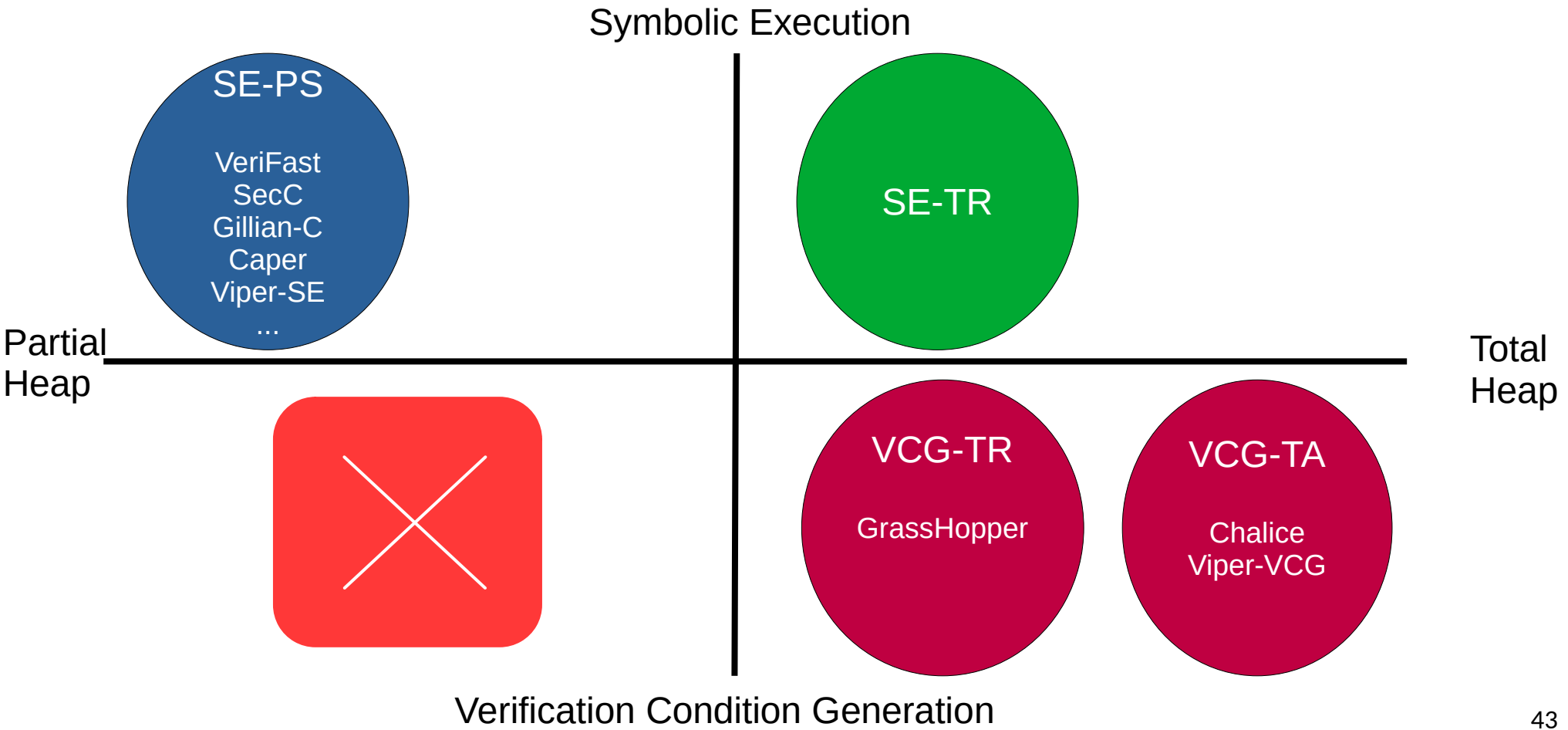


Store

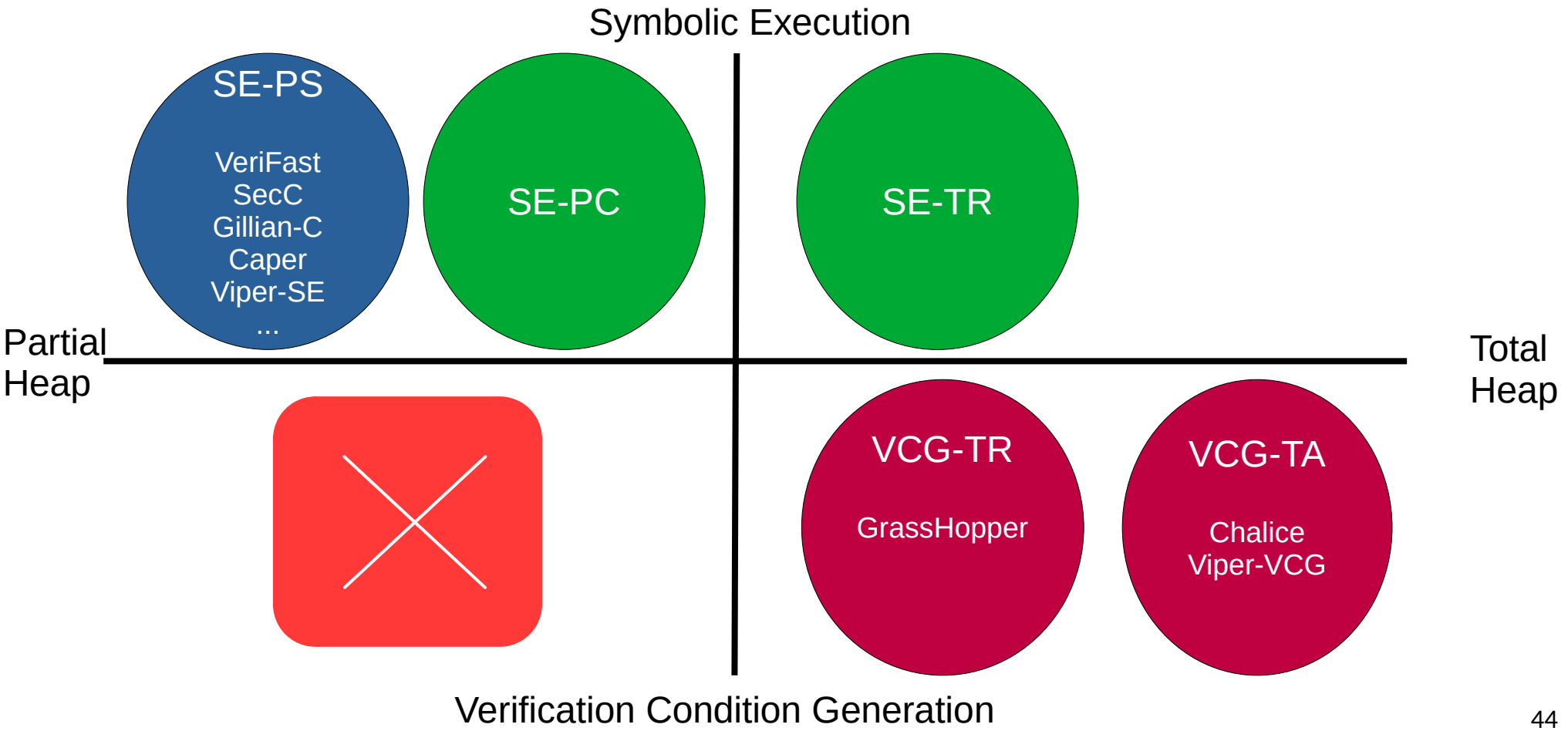
```
x → x0
y → 1
```

Heap: h

Algorithm Comparison



Algorithm Comparison



We implemented all five algorithms in Viper
and
benchmarked on large number of examples

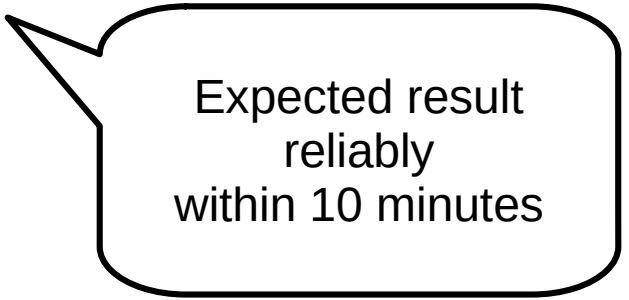
Benchmark Example Groups

- Source language verifiers
  - Rust, Java, Go, Python, CUDA, smart contracts
- Program logics
  - Concurrent separation logic, RSL, TaDA
- Properties
  - Information flow security, hyperproperties, reachability
- Applications
  - Verification tutorials, competitions, complex case studies

- 537 examples of various sizes overall

Completeness Results

Completeness Results



Completeness Results

	Σ	SE-PS	SE-PC	SE-TR	VCG-TR	VCG-TA
All	537	5.4	5.4	7.4	8.8	13.8
Ru_1	156	0.0	0.6	3.8	9.0	18.6
Ru_2	11	0.0	0.0	45.5	54.5	63.6
Go	11	18.2	0.0	18.2	0.0	0.0
Go_C	17	5.9	0.0	17.6	29.4	35.3
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9

Completeness Results

	Σ	SE-PS	SE-PC	SE-TR	VCG-TR	VCG-TA
All	537	5.4	5.4	7.4	8.8	13.8
Ru_1	156	0.0	0.6	3.8	9.0	18.6
Ru_2	11	0.0	0.0	45.5	54.5	63.6
Go	11	18.2	0.0	18.2	0.0	0.0
Go_C	17	5.9	0.0	17.6	29.4	35.3
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9

Algorithms

Completeness Results

Two different Rust verifiers

					-TR	VCG-TA
A					8.8	13.8
Ru_1	15	0.0	0.6	3.8	9.0	18.6
Ru_2	11	0.0	0.0	45.5	54.5	63.6
Go	11	18.2	0.0	18.2	0.0	0.0
Go_C	17	5.9	0.0	17.6	29.4	35.3
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9

Completeness Results

	Σ	SE-PS	SE-PC	SE-TR	VCG-T	
All	537	5.4	5.4	7.4	8.	
Ru_1	156	0.0	0.6	3.8	9.	
Ru_2	11	0.0	0.0	45.5		
Go	11	18.2	0.0	18.2		
Go_C	17	5.9	0.0	17.6	29.4	
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9

Percentage of Ru2 examples for which SE-TR was incomplete

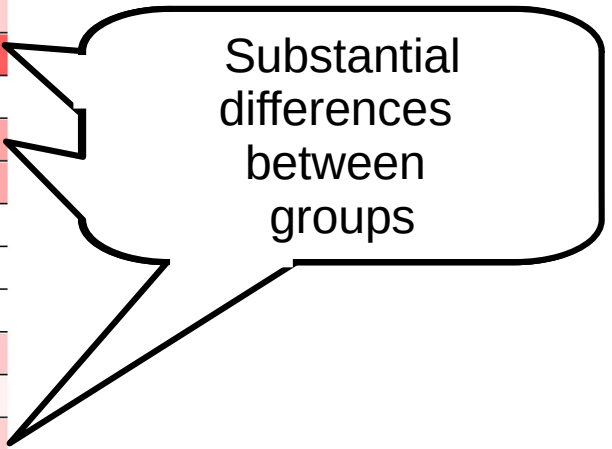
Completeness Results

	Σ	SE-PS	SE-PC	SE-TR	VCG-TR	VCG-TA
All	537	5.4	5.4	7.4	8.8	13.8
Ru_1	156	0.0	0.6	3.8	9.0	18.6
Ru_2	11	0.0	0.0	45.5	54.5	63.6
Go	11	18.2	0.0	18.2	0.0	0.0
Go_C	17	5.9	0.0	17.6	29.4	35.3
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9

SE-algorithms perform best overall

Completeness Results

	Σ	SE-PS	SE-PC	SE-TR	VCG-TR	VCG-TA
All	537	5.4	5.4	7.4	8.8	13.8
Ru_1	156	0.0	0.6	3.8	9.0	18.6
Ru_2	11	0.0	0.0	45.5	54.5	63.6
Go	11	18.2	0.0	18.2	0.0	0.0
Go_C	17	5.9	0.0	17.6	29.4	35.3
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9



Completeness Tendencies

	Σ	SE-PS	SE-PC	SE-TR	VCG-TR	VCG-TA
All	537	5.4	5.4	7.4	8.8	13.8
Ru_1	156	0.0	0.6	3.8	9.0	18.6
Ru_2	11	0.0	0.0	45.5	54.5	63.6
Go	11	18.2	0.0	18.2	0.0	0.0
Go_C	17	5.9	0.0	17.6	29.4	35.3
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9

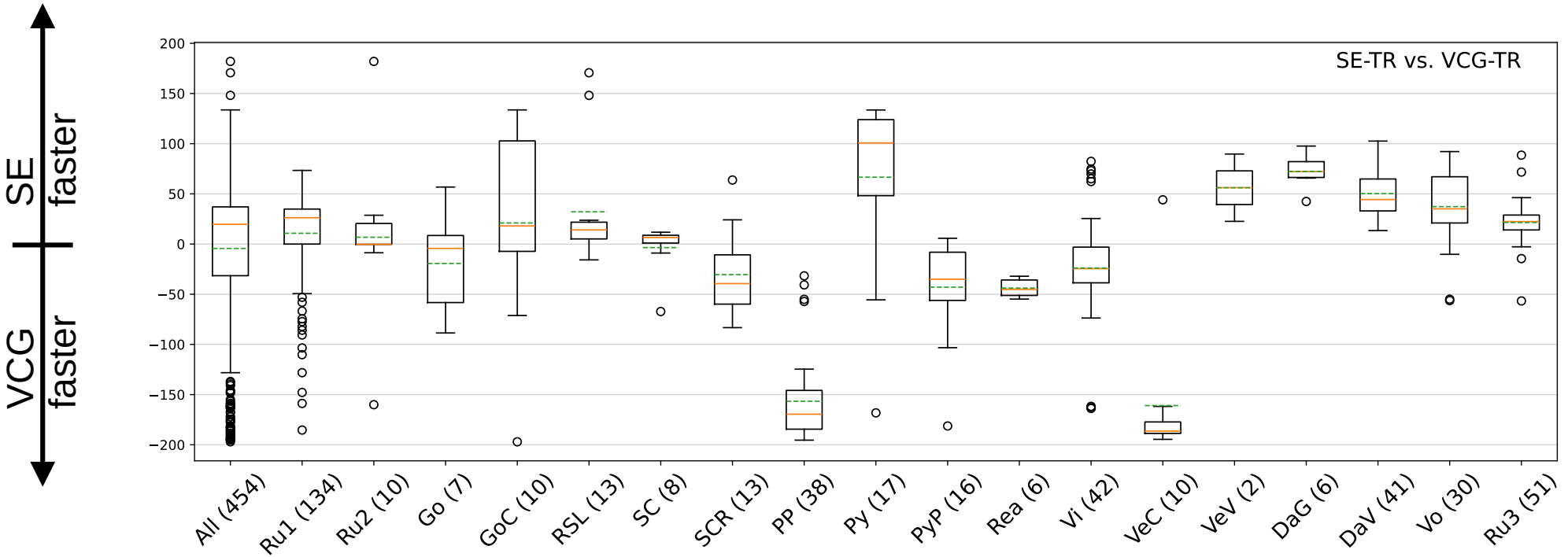
- Partial heap better:
  - Many simple heap operations
- Total heap better:
  - Heavy quantifier use
  - Lots of iterated separating conjunctions
- VCG struggles with large examples
- Future work: Predict best algorithm?

Portfolios

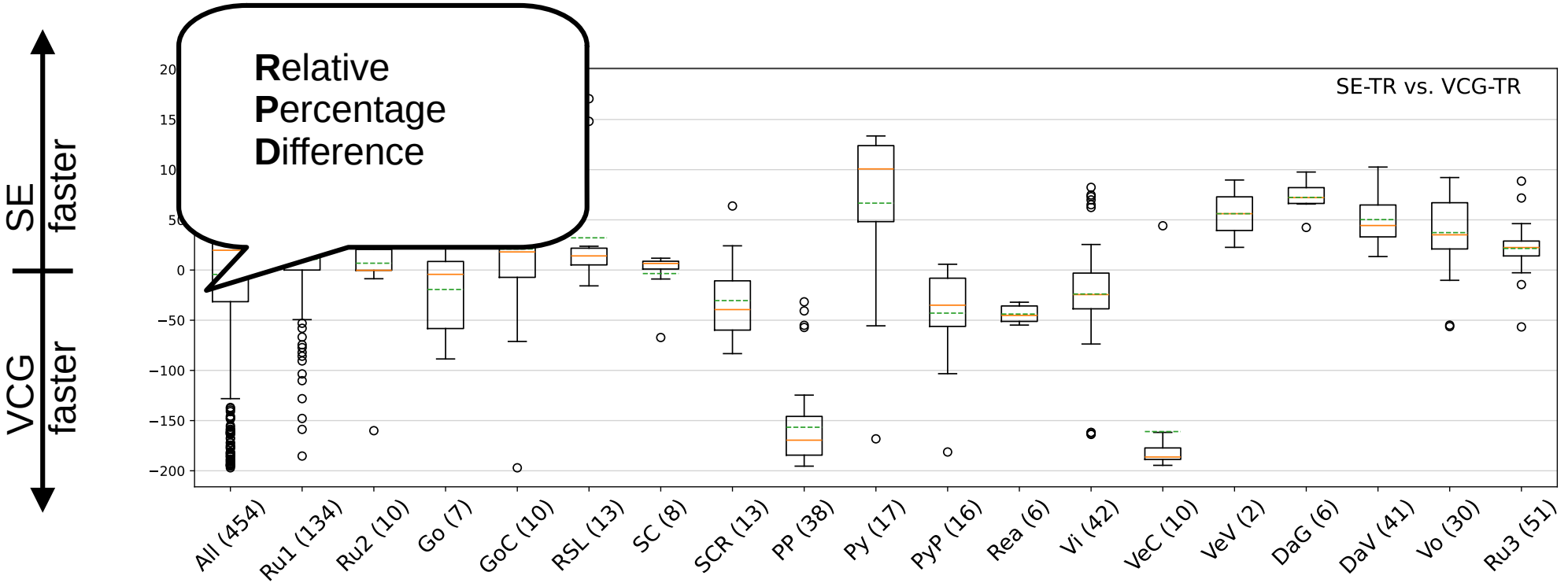
	Σ	SE-PS	SE-PC	SE-TR	VCG-TR	VCG-TA
All	537	5.4	5.4	7.4	8.8	13.8
Ru_1	156	0.0	0.6	3.8	9.0	18.6
Ru_2	11	0.0	0.0	45.5	54.5	63.6
Go	11	18.2	0.0	18.2	0.0	0.0
Go_C	17	5.9	0.0	17.6	29.4	35.3
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9

- No single algorithm can do everything
  - Portfolio recommended
- Four needed to cover all examples
- Three needed for all but one
  - SE-PS, SE-TR, any VCG
- SE-PS, VCG-TR for all but six
- SE-PS, SE-TR for all but ten
  - Easy to implement SE-TR in existing SE-PS

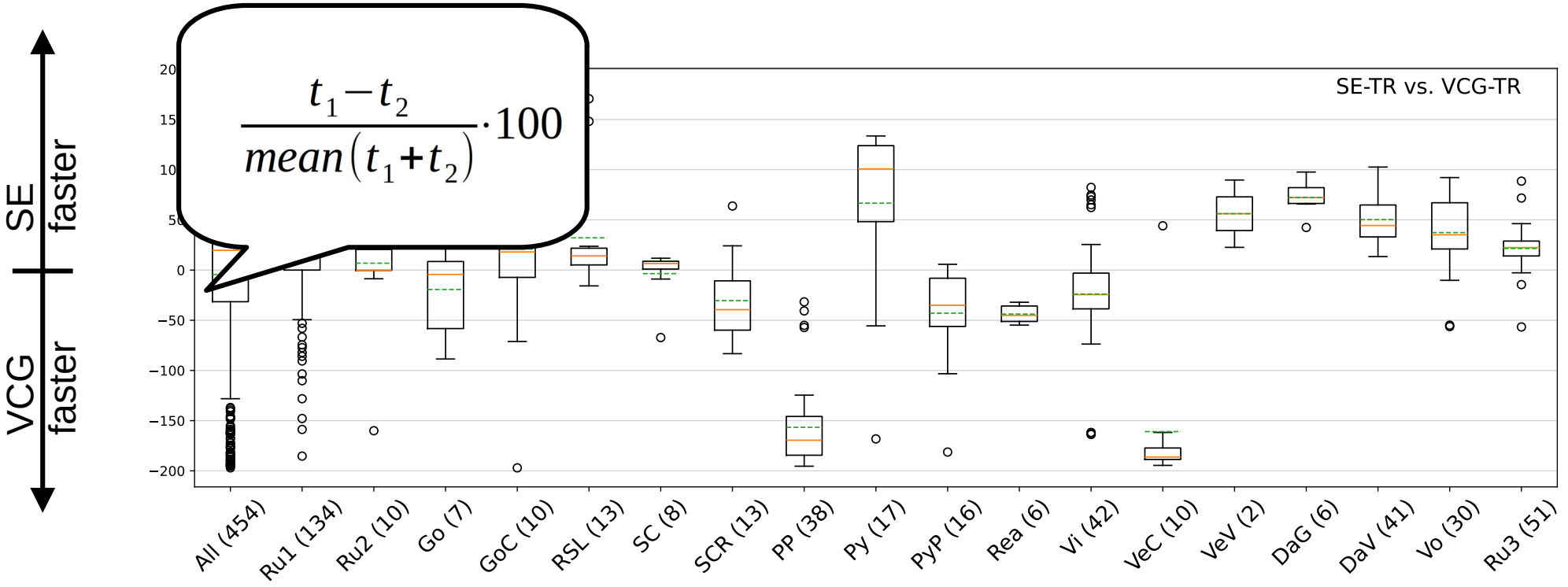
Performance Benchmark Example: SE vs. VCG with Total Heap



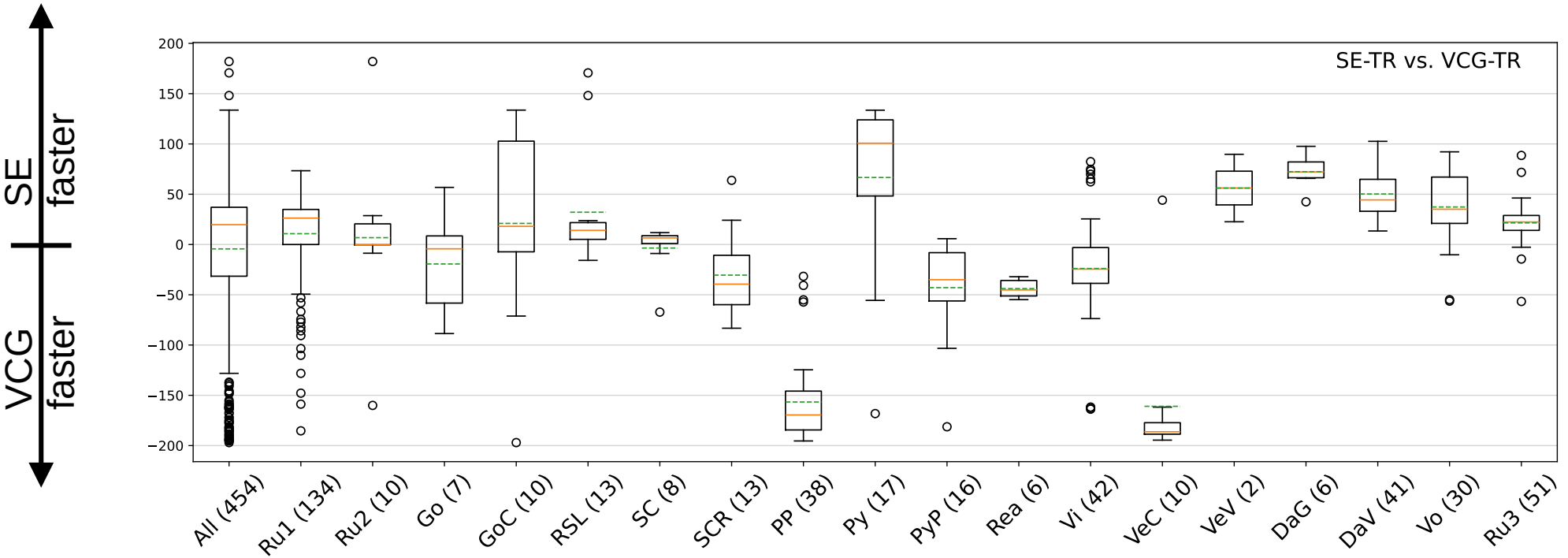
Performance Benchmark Example: SE vs. VCG with Total Heap



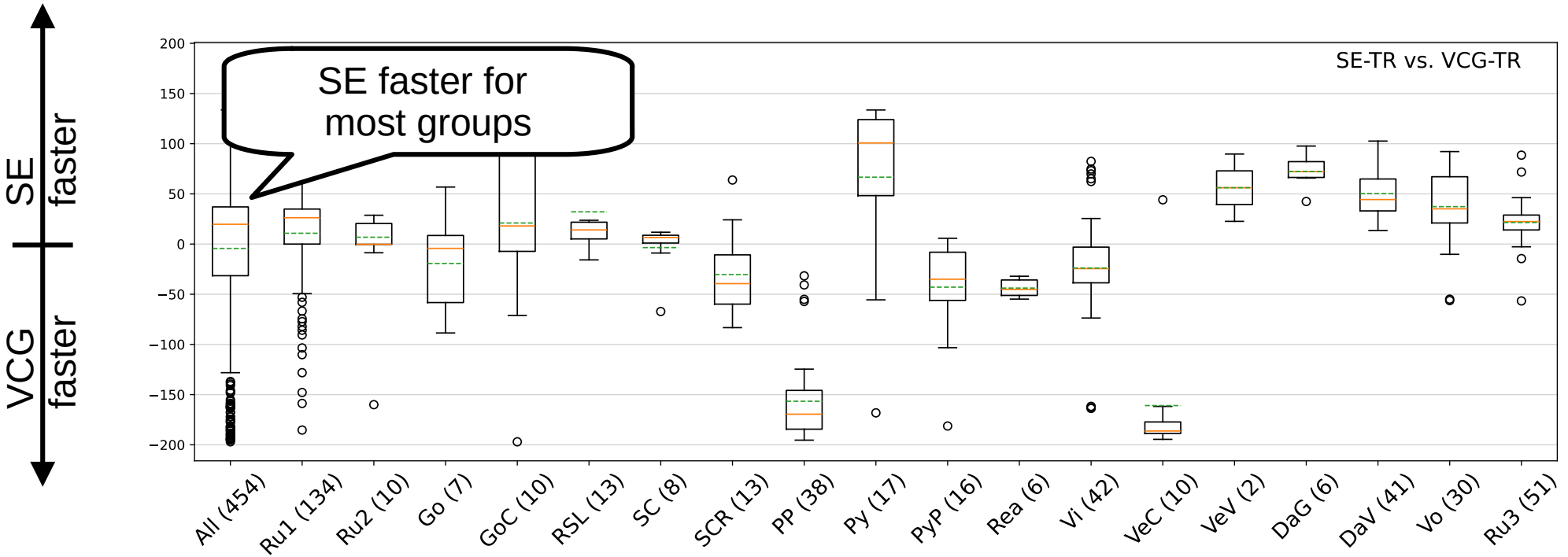
Performance Benchmark Example: SE vs. VCG with Total Heap



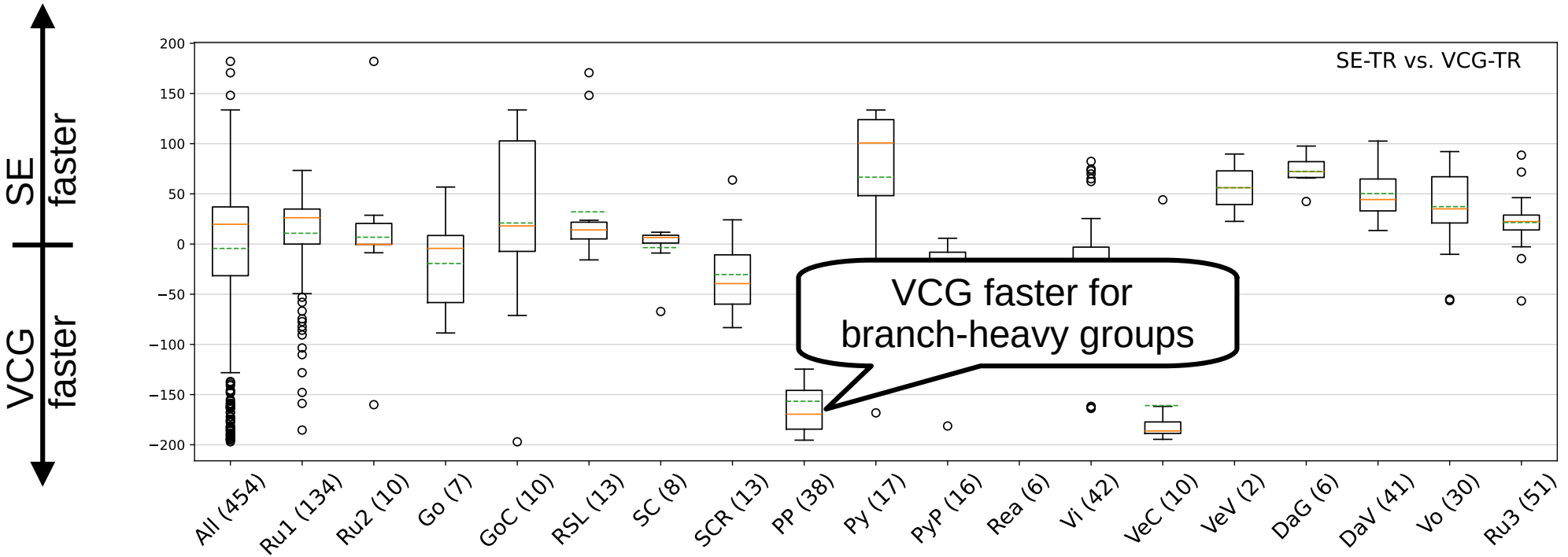
Performance Benchmark Example: SE vs. VCG with Total Heap



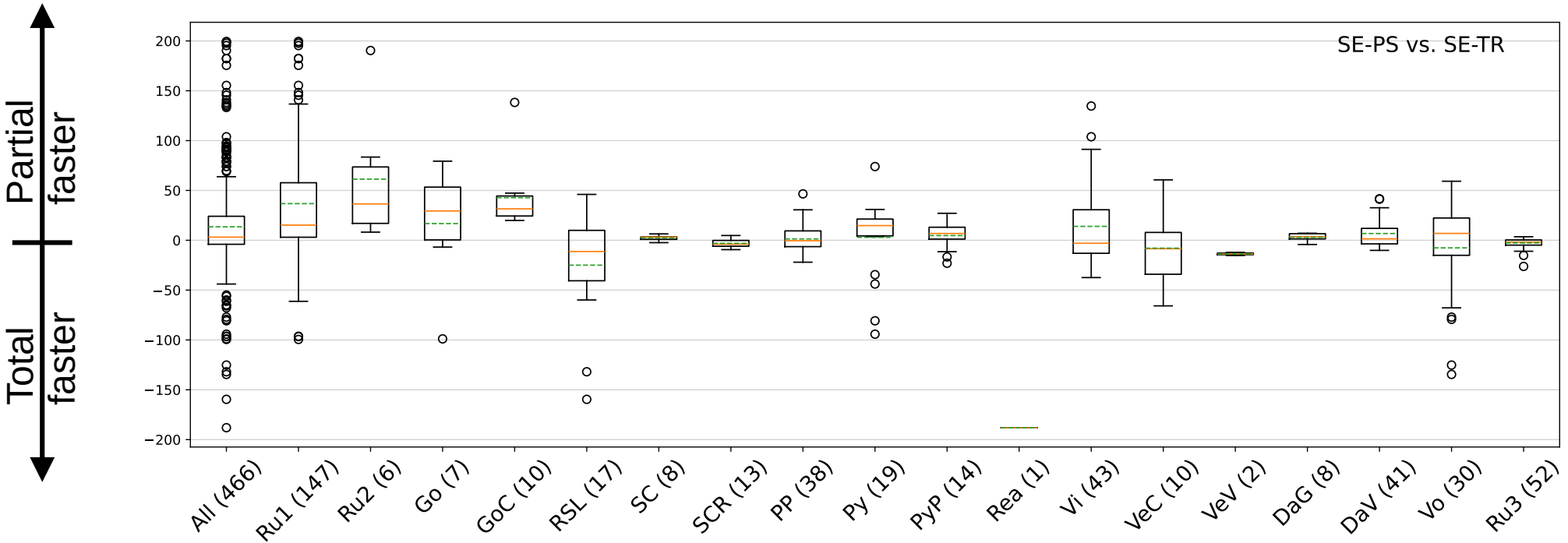
Performance Benchmark Example: SE vs. VCG with Total Heap



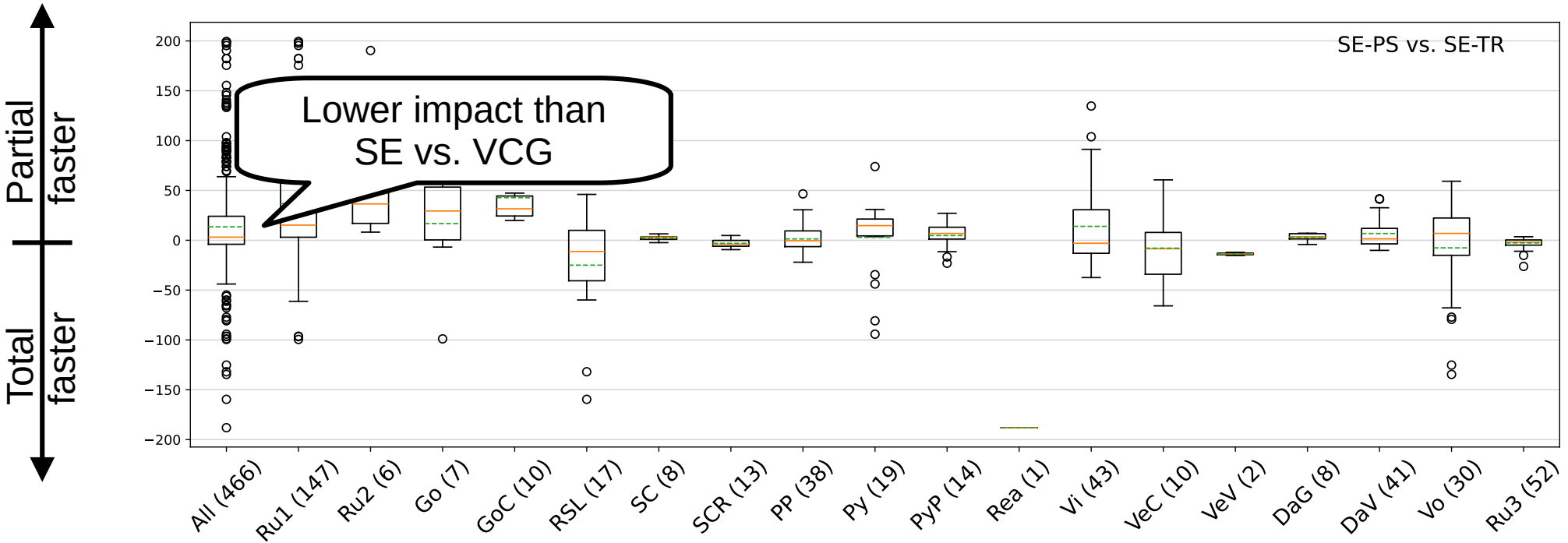
Performance Benchmark Example: SE vs. VCG with Total Heap



Performance Benchmark Example: SE with Partial vs. Total Heap

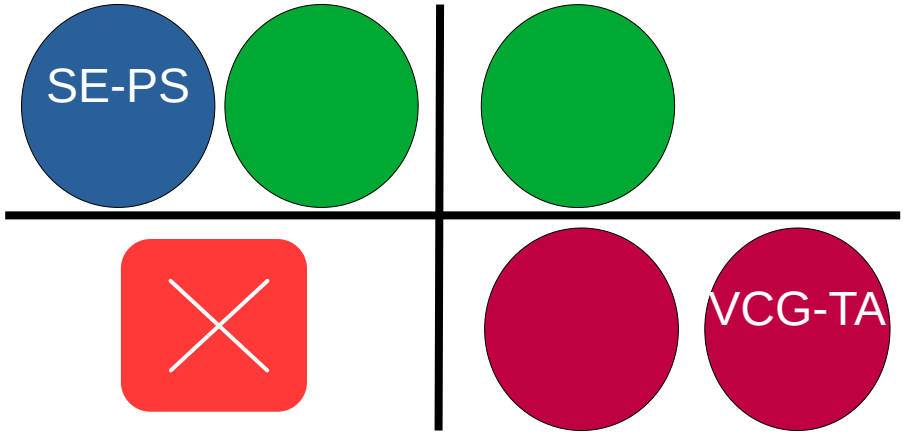


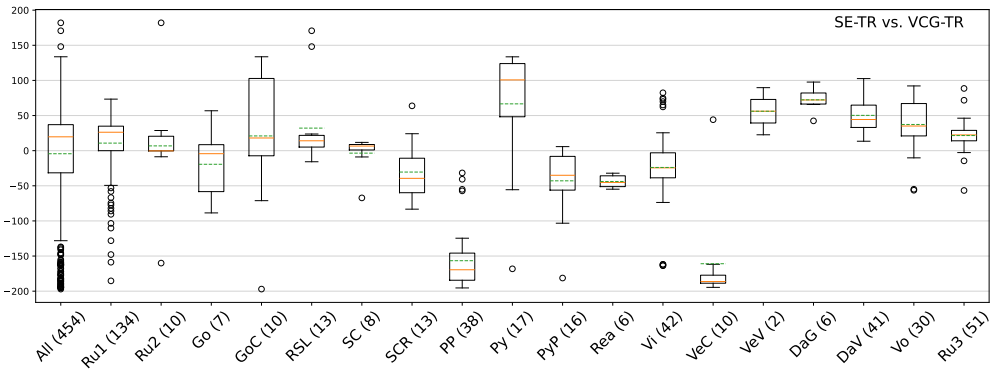
Performance Benchmark Example: SE with Partial vs. Total Heap



Conclusion

- SE + partial heap combinations perform best overall
  - Best overall completeness and performance
  - Most existing tools made a good choice
- Other algorithms are needed to cover all cases
  - Some example groups require total heaps
- Portfolio approach useful
- VCG-TR strictly better than VCG-TA
- Two new algorithms
  - SE-TR offers unique tradeoff
  - SE-PC is best overall along with SE-PS





	Σ	SE-PS	SE-PC	SE-TR	VCG-TR	VCG-TA
All	537	5.4	5.4	7.4	8.8	13.8
Ru_1	156	0.0	0.6	3.8	9.0	18.6
Ru_2	11	0.0	0.0	45.5	54.5	63.6
Go	11	18.2	0.0	18.2	0.0	0.0
Go_C	17	5.9	0.0	17.6	29.4	35.3
RSL	21	19.0	19.0	0.0	38.1	33.3
SC	8	0.0	0.0	0.0	0.0	0.0
SC_R	13	0.0	0.0	0.0	0.0	0.0
PP	38	0.0	0.0	0.0	0.0	0.0
Py	23	8.7	13.0	4.3	13.0	21.7
Py_P	18	16.7	11.1	11.1	5.6	5.6
Rea	16	93.8	93.8	37.5	12.5	18.8
Vi	46	2.2	2.2	4.3	4.3	8.7
Ve_C	19	0.0	10.5	42.1	10.5	10.5
Ve_V	5	0.0	0.0	20.0	20.0	0.0
Da_G	8	0.0	0.0	0.0	25.0	25.0
Da_V	41	0.0	0.0	0.0	0.0	0.0
Vo	34	2.9	2.9	11.8	0.0	20.6
Ru_3	52	0.0	0.0	0.0	1.9	1.9