CPAchecker: Configurable Software Verification

Dirk Beyer

Joint work with Erkan Keremoglu

SFU Simon Fraser University
BC, Canada

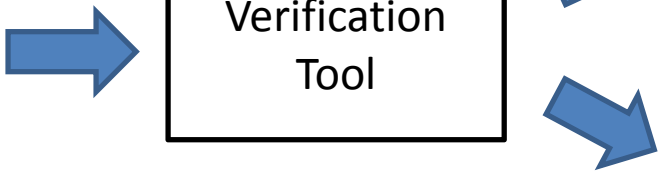
UNIVERSITÄT PASSAU

Automatic Software Verification

C program

```
int main() {
  int a = foo();
  int b = bar(a);

  assert(a == b);
}
```

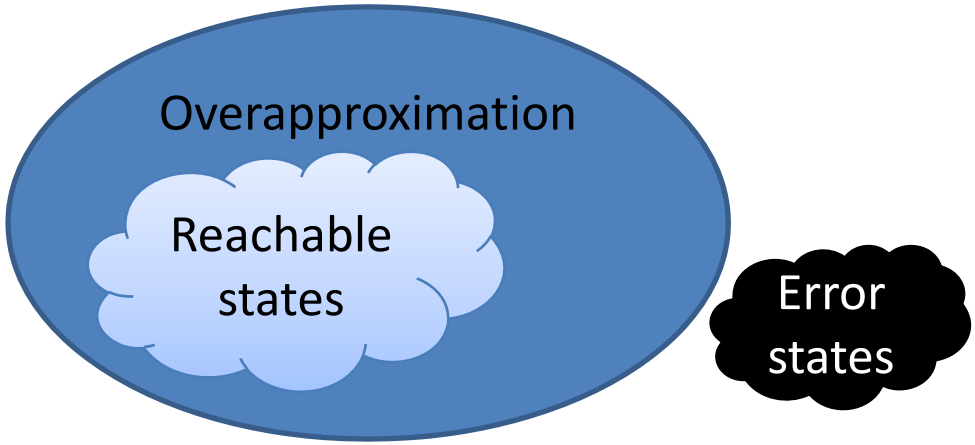


Verification Tool

SAFE
i.e. assertions cannot be violated

UNSAFE

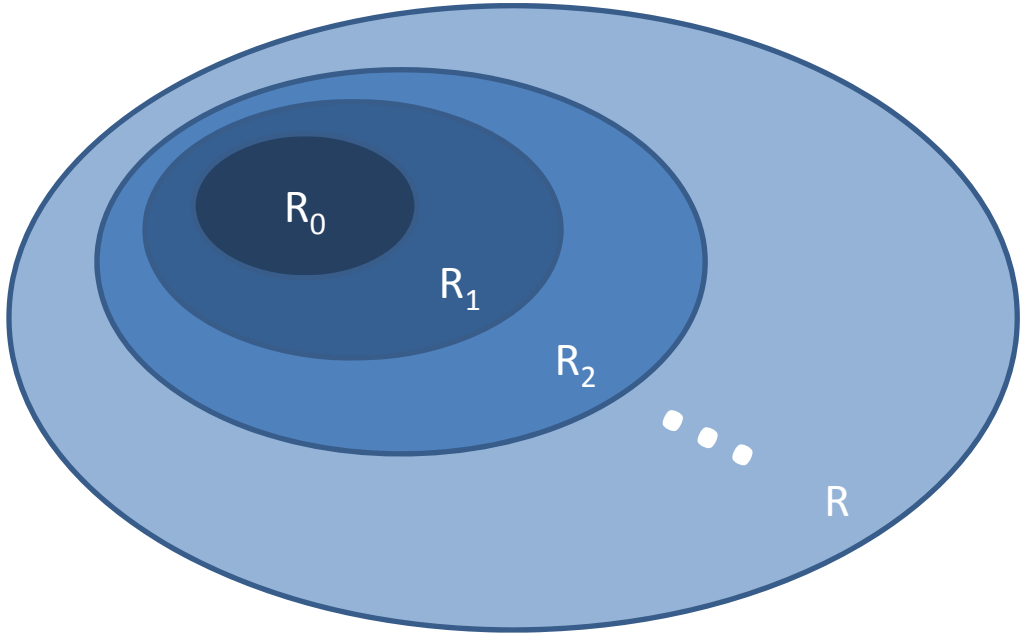
General method:
Create an overapproximation of the program states



Software Verification by Model Checking

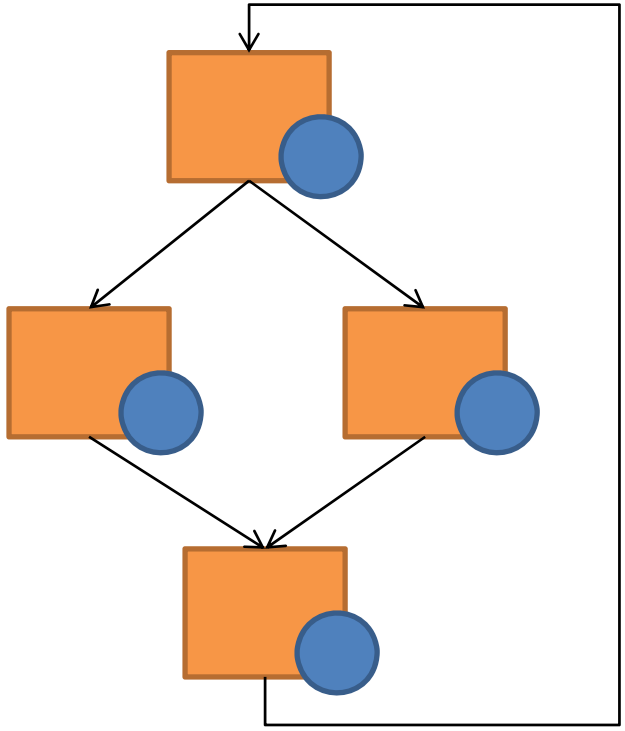
[Clarke/Emerson, Sifakis 1981]

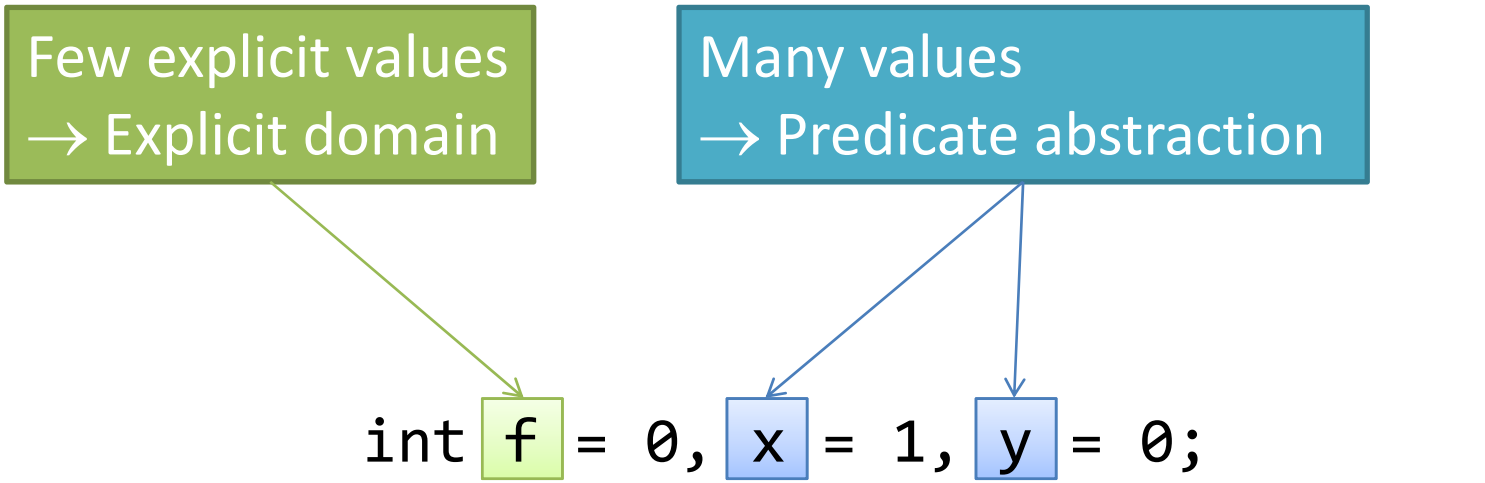
Iterative fixpoint (forward) post computation



Software Verification by Data-flow Analysis

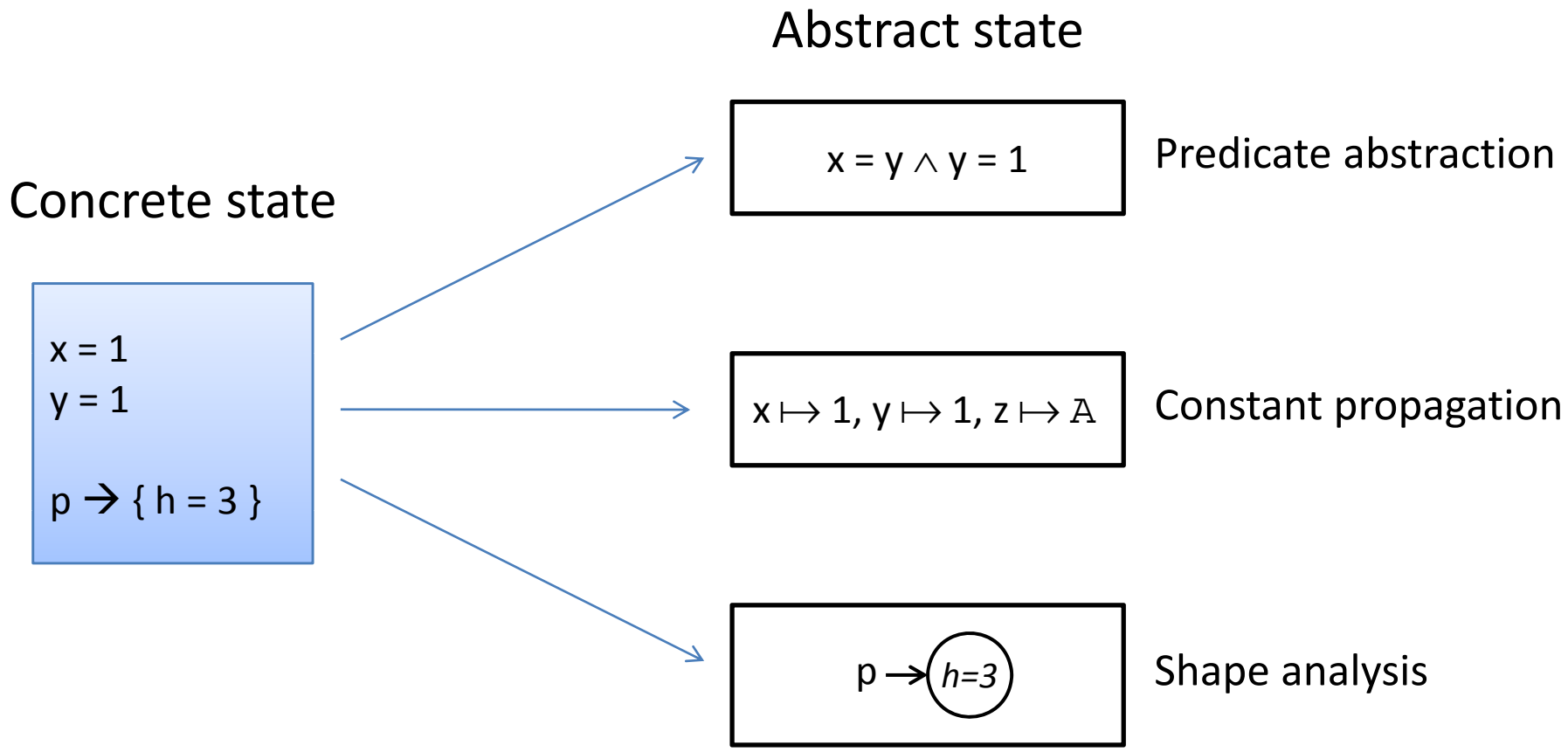
Fixpoint computation on the CFG






```
int f = 0, x = 1, y = 0;
while (x > 0) {
  if (f == 0) { x = 50; f = 1; }
  else        { x--; y++; }
}
assert(y == 50);
```

Abstraction



Motivation

- BLAST is one of the most important tools because it proved many ideas useful
- Monolithic, difficult to extend, decr. entropy

Need for:

- Component-based verification platform
- Flexible algorithm for adjustable precision
- Benchmark environment

Software Model Checking

Reached, *Frontier* := { e_0 }

while *Frontier* $\neq \emptyset$ do

    remove *e* from *Frontier*

    for each *e'* $\in$ **post**(*e*) do

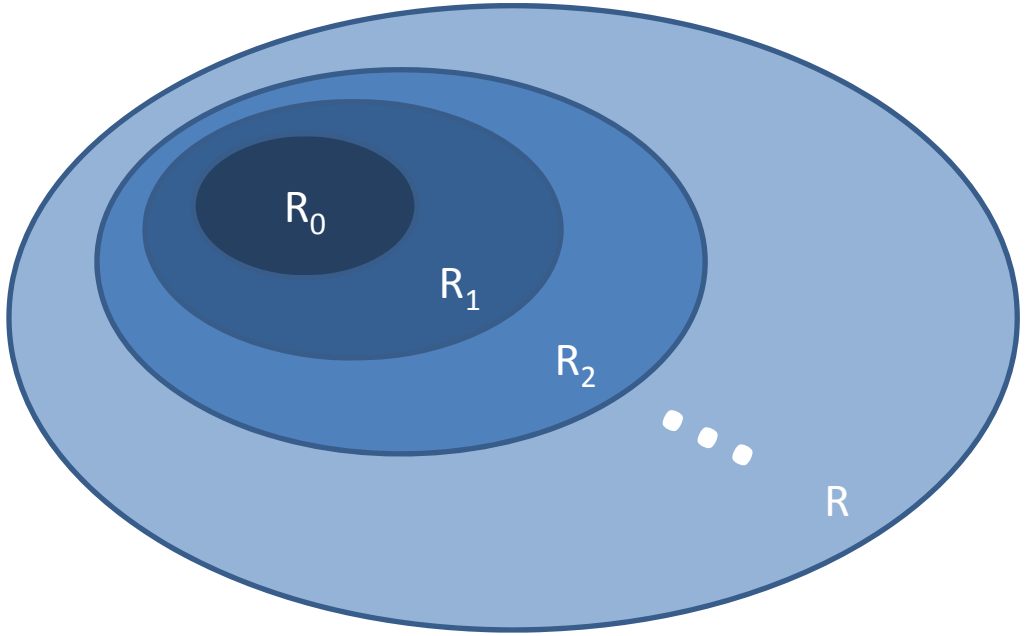
        if $\neg$ **stop**(*e'*, *Reached*) add *e'* to *Reached*, *Frontier*

return *Reached*

Software Verification by Model Checking

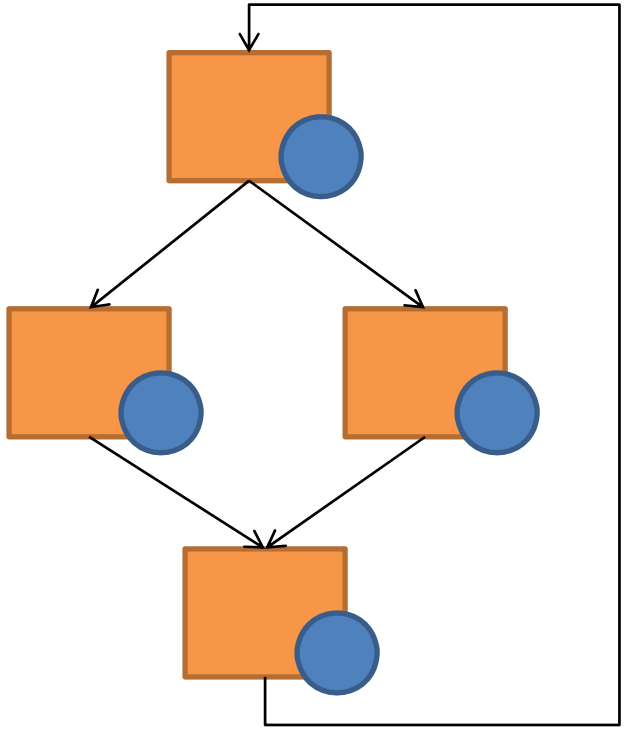
[Clarke/Emerson, Sifakis 1981]

Iterative fixpoint (forward) post computation



Software Verification by Data-flow Analysis

Fixpoint computation on the CFG



Configurable Program Analysis [CAV 2007]

Reached, *Frontier* := { e_0 }

while *Frontier* $\neq \emptyset$ do

  remove *e* from *Frontier*

  for each *e'* $\in$ **post**(*e*) do

    for each *e''* $\in$ *Reached* do

      e''_{new} := **merge**(*e'*, *e''*)

      if $e''_{new} \neq e''$ then

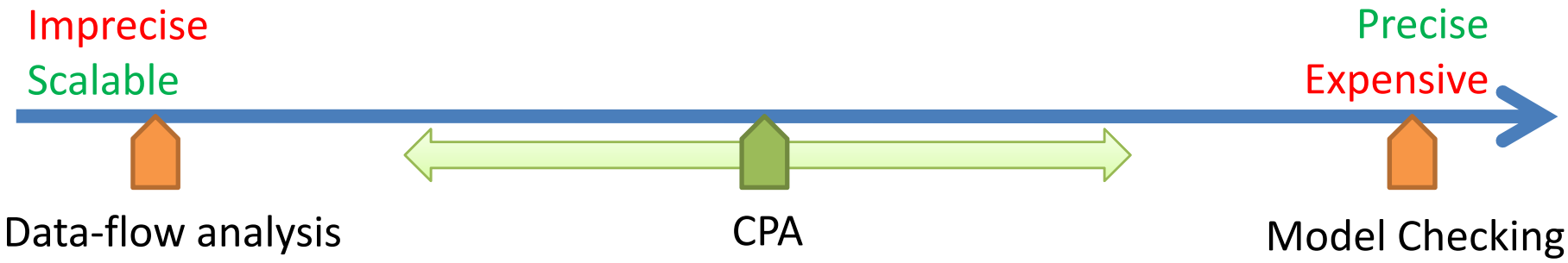
        replace *e''* in *Reached*, *Frontier* by e''_{new}

    if $\neg$ **stop**(*e'*, *Reached*) add *e'* to *Reached*, *Frontier*

return *Reached*

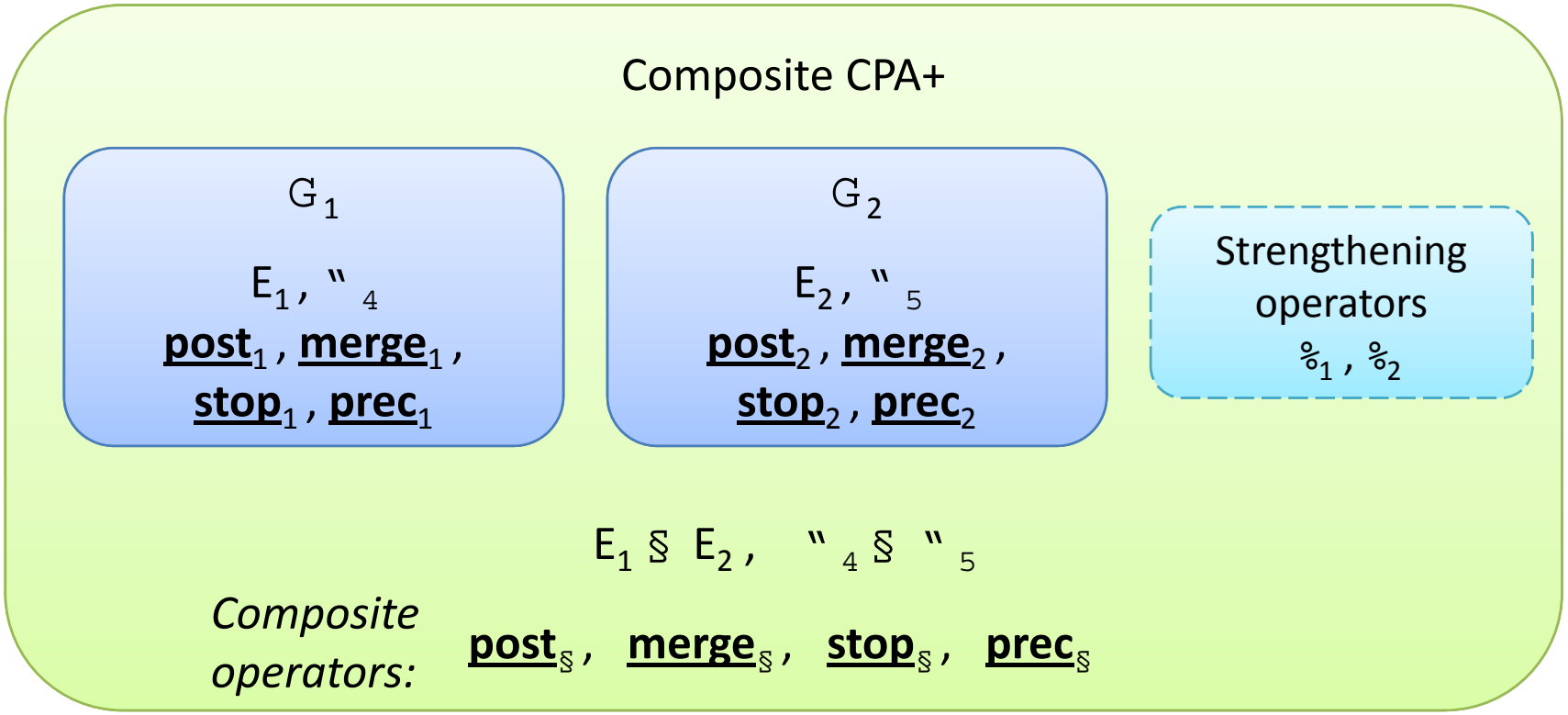
Configurable Program Analysis

- Better combination of abstractions
  ➔ Configurable Program Analysis [CAV07]

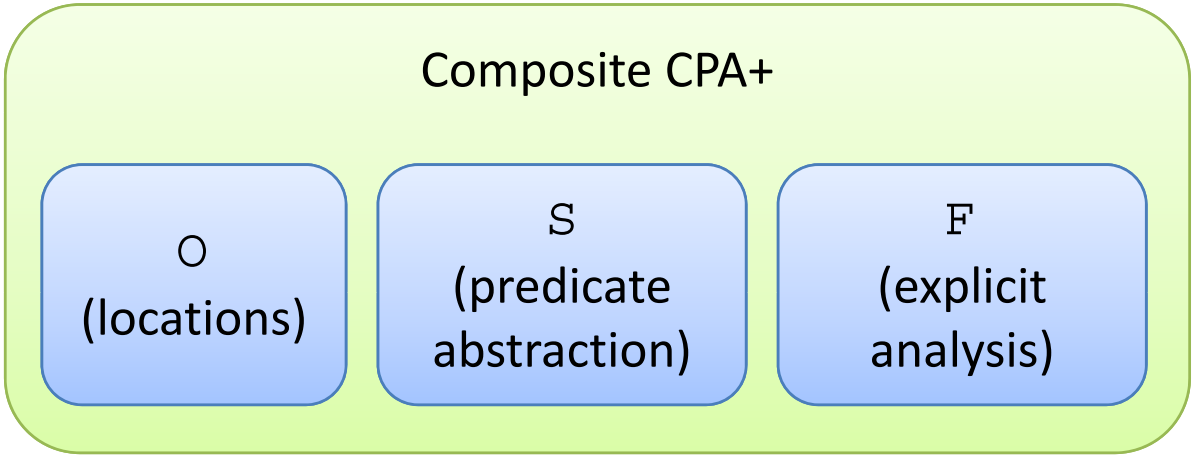


Unified framework that enables intermediate algorithms

Composite CPA+



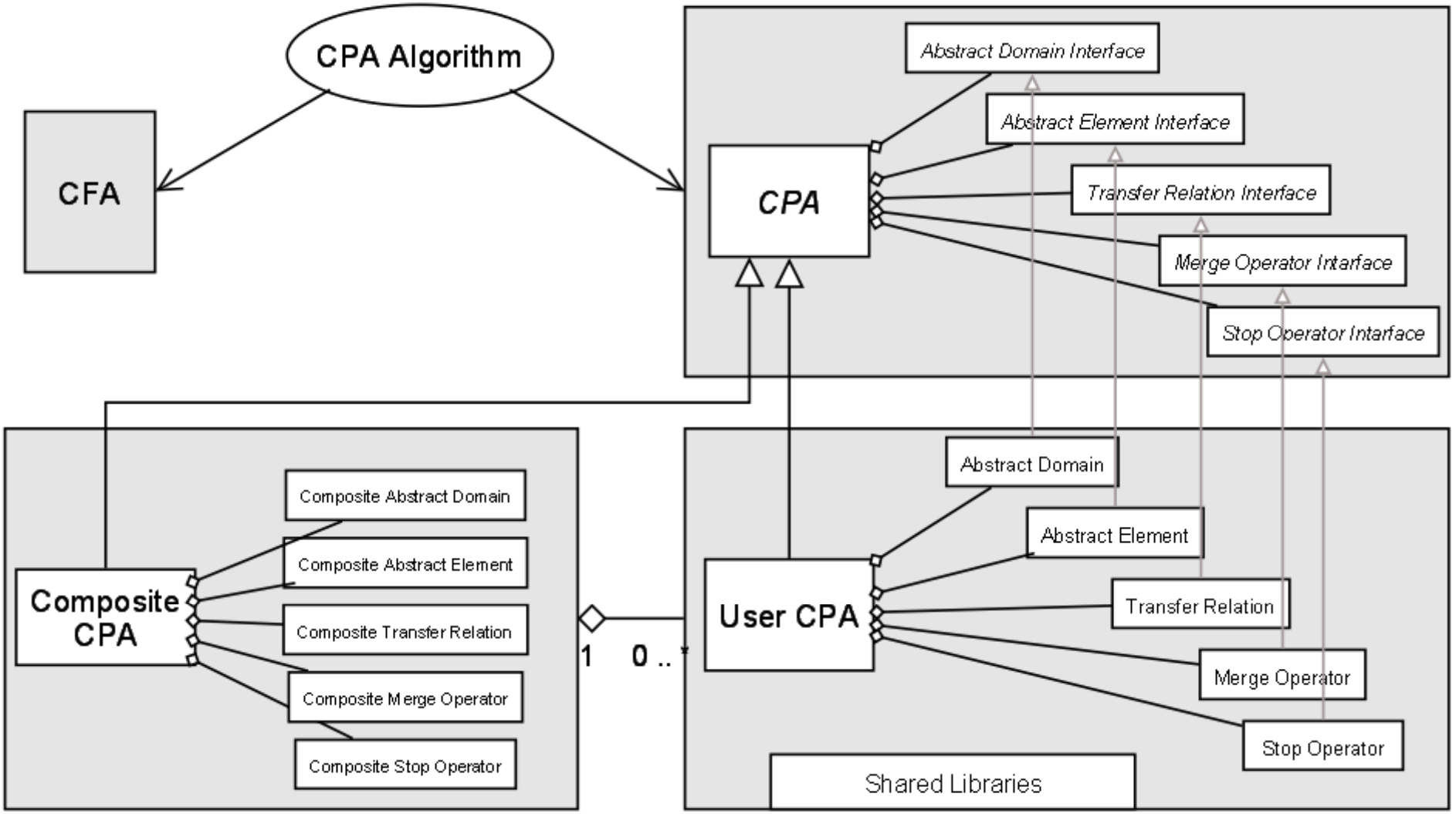
Example: Predicate Abstraction + Explicit



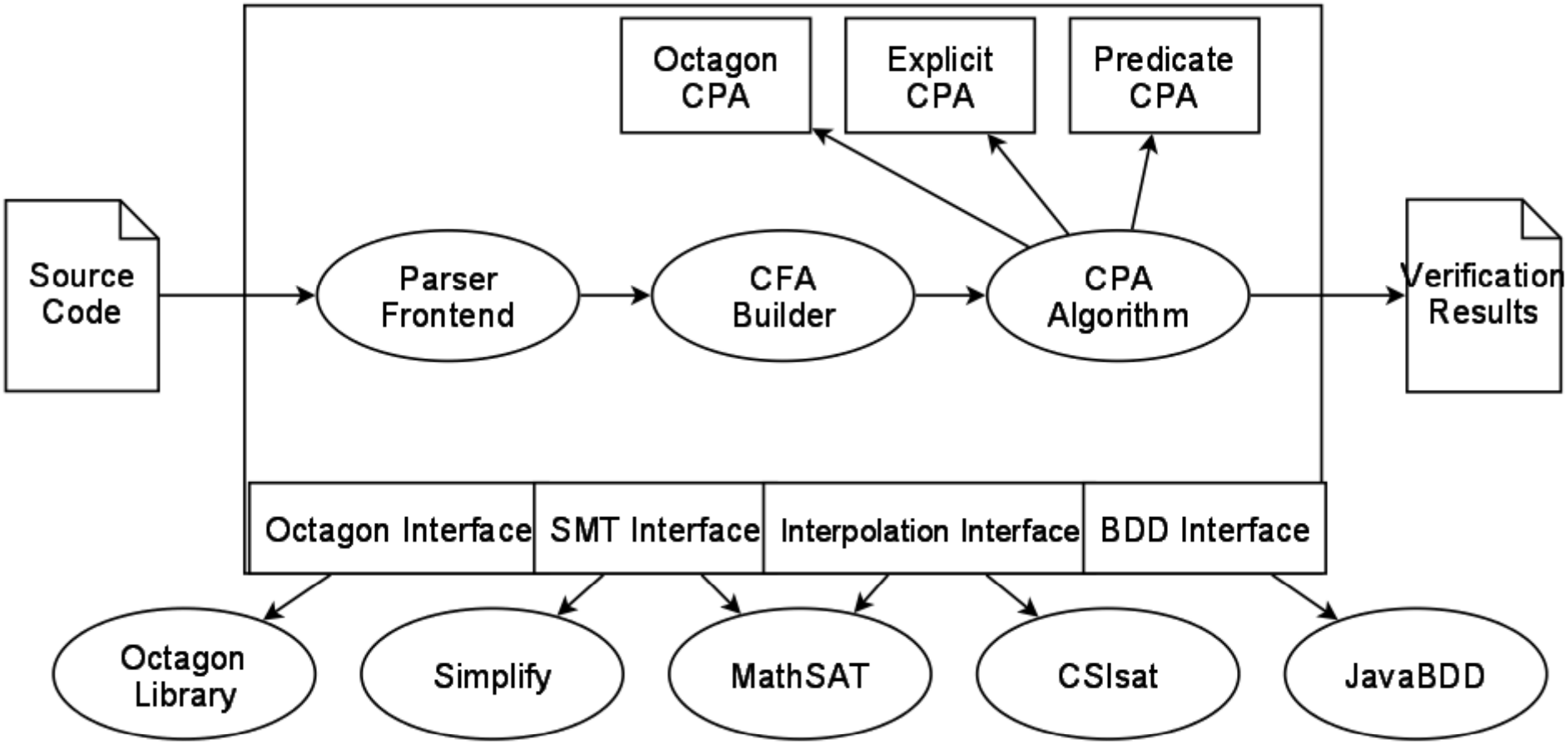
Example of composite abstract element:

(6 , x > 0 – x = y , { i ↦ 2, x ↦ A , y ↦ A })

CPAchecker -- Design



CPAchecker -- Architecture and Flow



Conclusion

- (Beginning of) New tool: CPAchecker

- Component-based verification platform
- Flexible algorithm for adjustable precision

- Easy to extend
- Potential to serve as a widely used platform