Metamorph: Synthesizing Large Objects with Dafny APIs

Aleksandr Fedchin, **Alexander Bai**, Jeff Foster

Dafny

- Verification-aware language that allows writing specification in first order logic.
- Check the program w.r.t. the specification with SMT solvers
- UNSAT means verified, a satisfying assignment will give back an counterexample

Dafny is Used for Critical Systems

AWS Encryption SDK &
Authorization Engine

Dafny EVM

Project Everest

Ironclad/Ironfleet

Motivation

- Existing tools mostly restrict themselves to functional code
- Many synthesis tools have a max number of methods
- Synthesis tools usually require execute candidate solutions
- Dafny allows **full** specification of API methods, including of side-effects

Motivating Example: Social Network API

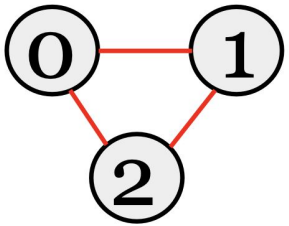
```
datatype User = User(name:nat,
      friends:set<nat>,
      incomingRequests:set<nat>)

class SocialNetwork {
  ghost var users:map<nat,User>

  constructor()...
  method AddUser(id:nat)...
  method RequestConnect(from:nat, to:nat) ...
  method RemoveUser(id:nat)...
  method RequestResponse(from:nat, to:nat, response:bool)...
}
```

Method Specification Example

```
method AddUser(id:nat)
  requires id !in users
  modifies this
  ensures users == old(users)[id := User(id, {}, {})]
{
    users := users[id := User(id, {}, {})]
}
```

Example Synthesis Query

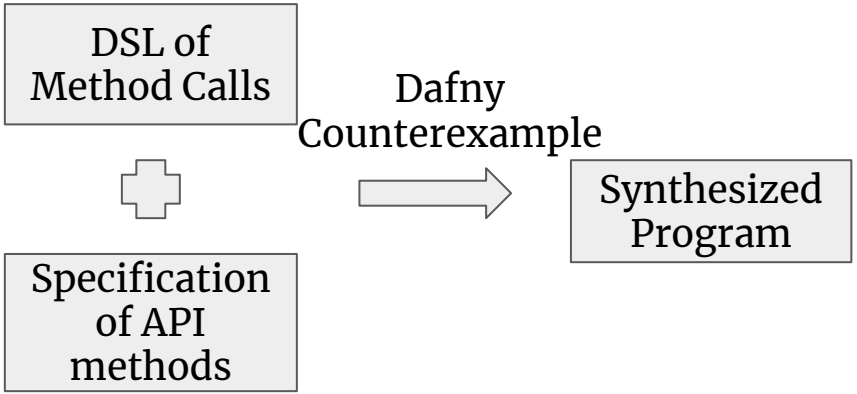
```
predicate Target(s:SocialNetwork)
  reads db
{
  s.users.Keys == {0, 1, 2} &&
  0 in s.users[1].friends &&
  1 in s.users[2].friends &&
  2 in s.users[0].friends
}
```




```
static method Solution()
  returns (s:SocialNetwork)
  ensures Target(result)
{
  s := new SocialNetwork();
  s.AddUser(1);
  s.AddUser(0);
  s.RequestConnect(1, 0);
  s.RequestResponse(0, 1, true);
  s.AddUser(2);
  s.RequestConnect(1, 2);
  s.RequestResponse(2, 1, true);
  s.RequestConnect(2, 0);
  s.RequestResponse(0, 2, true);
}
```

Solution 1: Using Dafny Directly

We can use Dafny counterexamples to synthesize the goal directly.




```
method Synthesize(s:seq<MethodCall>) {
  var network := new SocialNetwork();
  Execute(s, network);
  if (Target(network)) {
    assert false;
  }
}
```

Problem: The excessive use of quantifiers makes this approach impractical

What if we reason about one method at a time?

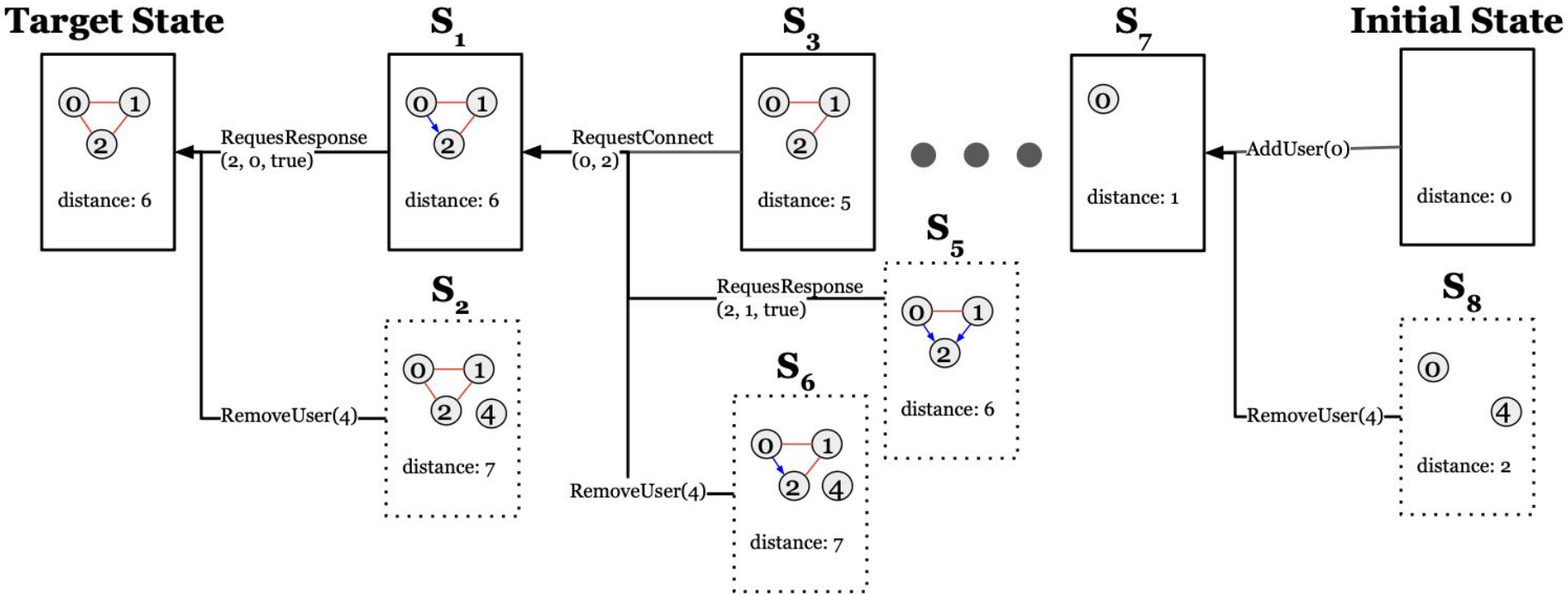
Backward Synthesis by Counterexample

- Consider the following assertion:

```
method InferPreviousState(s: SocialNetwork, arg: nat)
  modifies receiver {
    assume !Target(s);
    assume arg in s.users;
    s.RemoveUser(arg);
    assert !Target(s);
}
```

- The counterexample will give us the the previous state as a conjunction of constraints

Backward Synthesis as Graph Search



Distance Metric

- Counterexamples are variable assignment, so they represent object states as conjunctions of constraints.

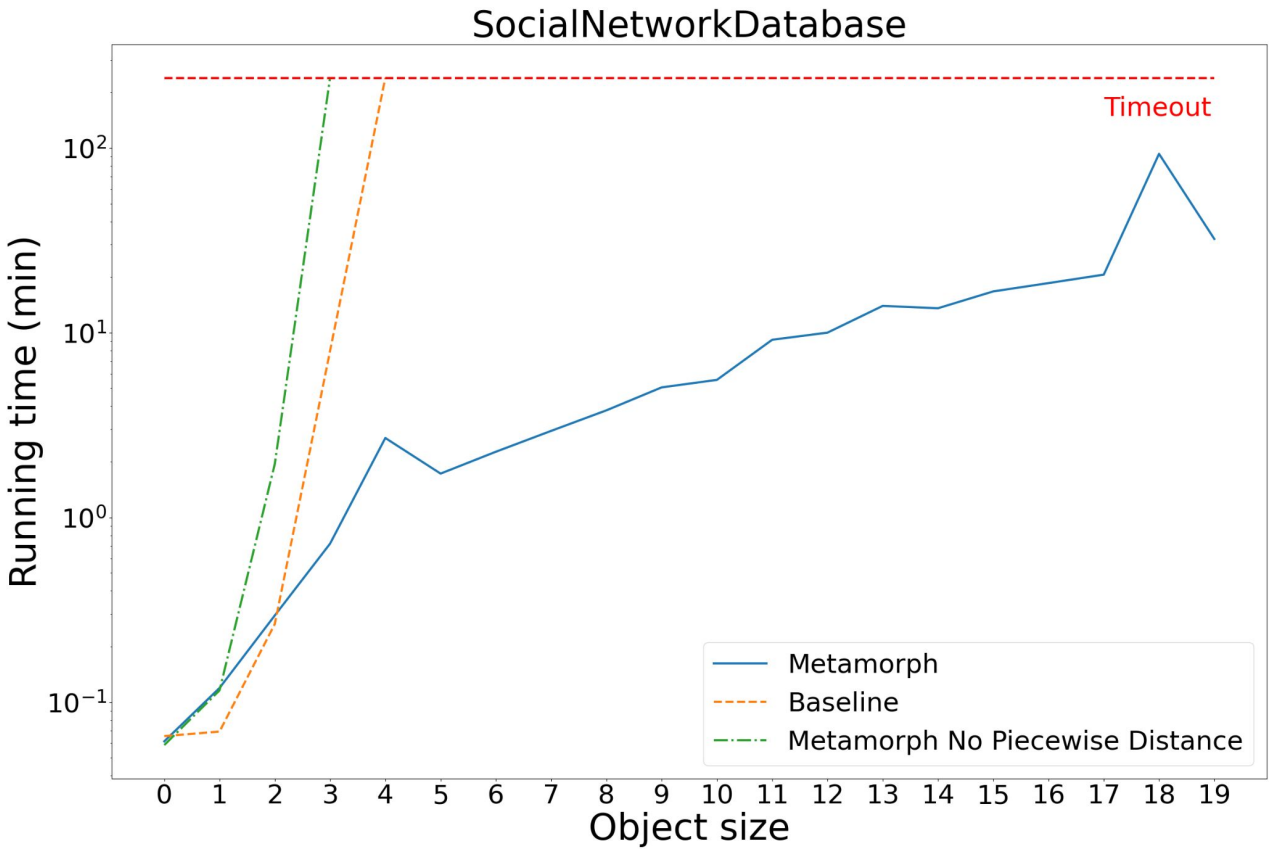
- We abstract away concrete values in these constraints...

```
0 in users && 1 in users <=> X(a) && X(b) && a != b
```

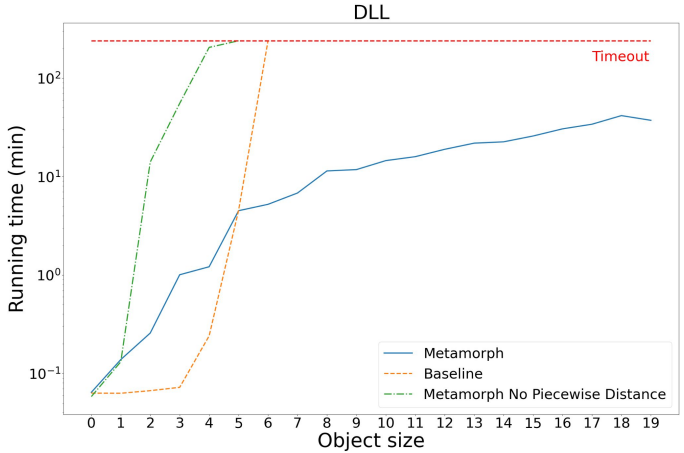
- ...and reason about how each API method interacts with each symbolic constraint.

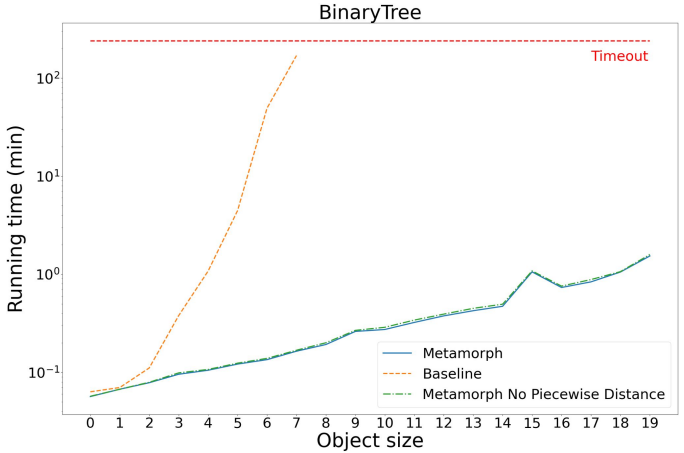
```
X(a) -> RemoveUser(a) -> ¬X(a)
```

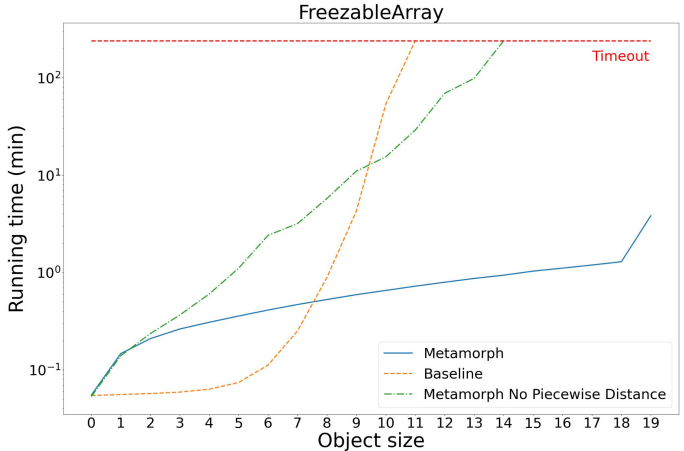
Evaluation

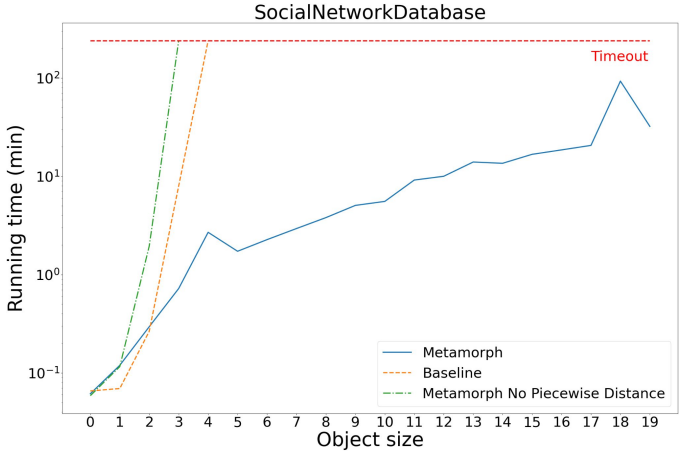

SocialNetworkDatabase
Timeout
Running time (min)
10^2
10^1
10^0
10^{-1}
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Object size
Metamorph
Baseline
Metamorph No Piecewise Distance

Evaluation


DLL
Timeout
Running time (min)
Object size
Metamorph
Baseline
Metamorph No Piecewise Distance


BinaryTree
Timeout
Running time (min)
Object size
Metamorph
Baseline
Metamorph No Piecewise Distance


FreezableArray
Timeout
Running time (min)
Object size
Metamorph
Baseline
Metamorph No Piecewise Distance


SocialNetworkDatabase
Timeout
Running time (min)
Object size
Metamorph
Baseline
Metamorph No Piecewise Distance

Future Work

- Integrate with Dafny Test Generation framework.
- Synthesizing programs with non-linear control flow.
- Combining forward and backward search

Summary

- We introduced an approach of using counterexamples to “concretize” states

- We used a distance metric to guide the backward synthesis