

Lecture 3: Solver-Aided Programming II

Yu Feng
Spring 2021

Summary of previous lecture

- The first paper review is out
- The classical way for using solvers
- Solver-aided programming I
- Rosette constructs

A programming model that integrates solvers into the language, providing constructs for program verification, synthesis, and debugging.

Solver-aided programming

```
p(x) {  
  v = 12
```

```
p(x) {  
  v = ??
```

```
  ...
```

```
}
```

```
assert safe(x, p(x))
```

Find an input on which the program fails.

Localize bad parts of the program.

Find values that repair the failing run.

Find code that repairs the program.

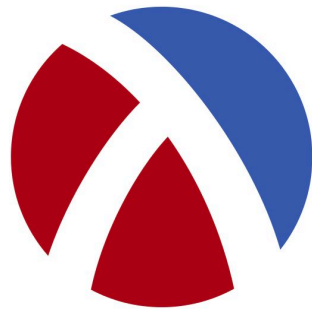


Rosette constructs



Rosette

=



Racket

+

```
(define-symbolic id type)
(define-symbolic* id type)
```

**symbolic
values**

```
(assert expr)
```

assertions

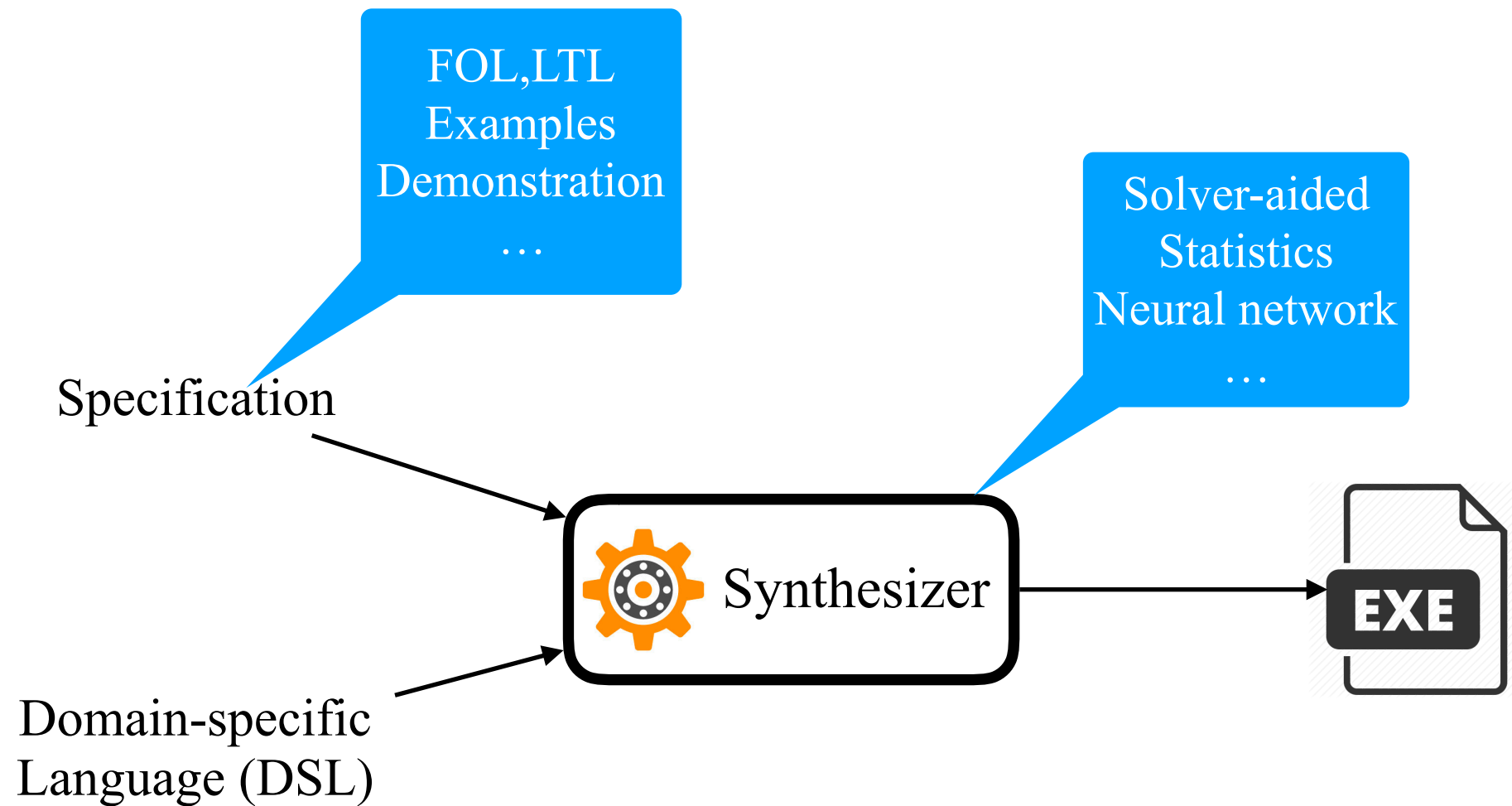
```
(verify expr)
(debug [type ...+] expr)
(solve expr)
(synthesize
  #:forall expr
  #:guarantee expr)
```

queries

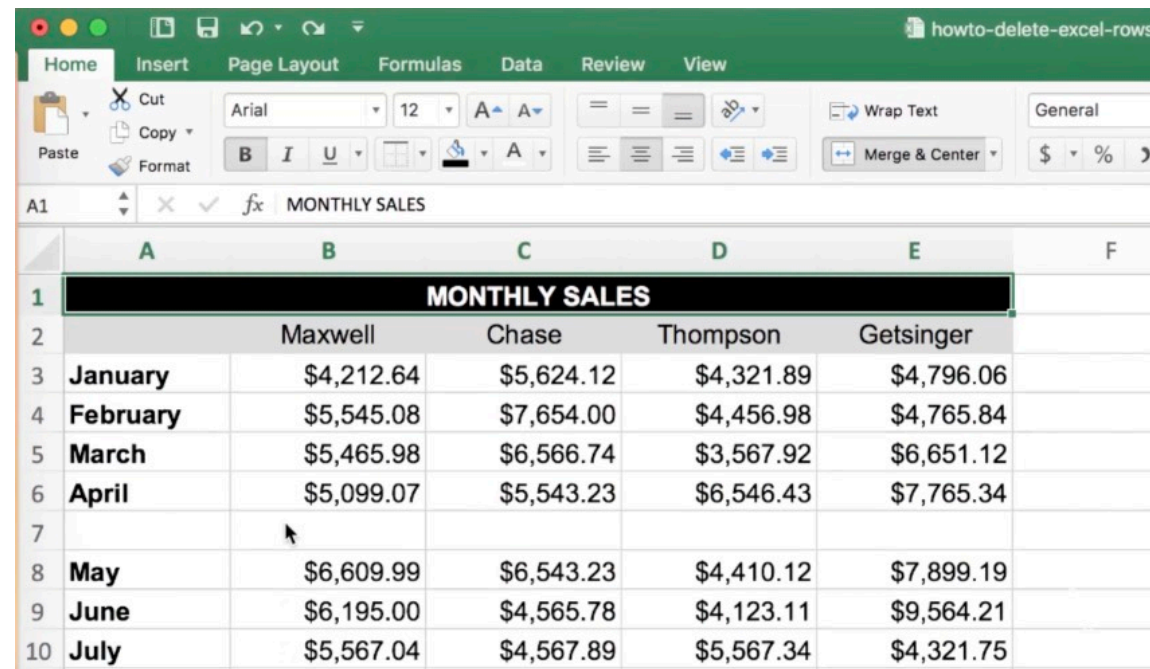
Outline of this lecture

- The spectrum of program synthesis
- Solver-aided programming II (synthesis)
- Program synthesis via conflict-driven learning

What is program synthesis



Program-by-example



	A	B	C	D	E	F
1	MONTHLY SALES					
2		Maxwell	Chase	Thompson	Getsinger	
3	January	\$4,212.64	\$5,624.12	\$4,321.89	\$4,796.06	
4	February	\$5,545.08	\$7,654.00	\$4,456.98	\$4,765.84	
5	March	\$5,465.98	\$6,566.74	\$3,567.92	\$6,651.12	
6	April	\$5,099.07	\$5,543.23	\$6,546.43	\$7,765.34	
7						
8	May	\$6,609.99	\$6,543.23	\$4,410.12	\$7,899.19	
9	June	\$6,195.00	\$4,565.78	\$4,123.11	\$9,564.21	
10	July	\$5,567.04	\$4,567.89	\$5,567.34	\$4,321.75	

Two minutes tour to the FlashFill system



N Ph.D. students



Sumit Gulwani

<https://www.youtube.com/watch?v=lCVOmWdy1Hc>

Program-by-demonstration



One minute tour to the Helena system



Sarah Chasins



Ras Bodik

<https://tinyurl.com/y35936gr>

<http://helena-lang.org/>

Program-by-natural-language



Amazon Alexa



SQLizer



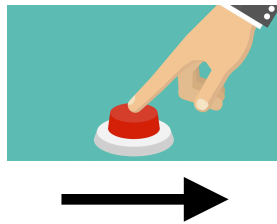
IFTTT

Yaghmazadeh, Navid, et al. SQLizer: query synthesis from natural language. OOPSLA 2017.

Quirk, Chris, Raymond Mooney, and Michel Galley. "Language to code: Learning semantic parsers for if-this-then-that recipes." ACL 2015.

Program-by-types

```
quickSort(arr[], low, high)
{
}
```



```
quickSort(arr[], low, high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[pi] is now
           at right place */
        pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1); // Before pi
        quickSort(arr, pi + 1, high); // After pi
    }
}
```

Five minutes tour to the Hunter system

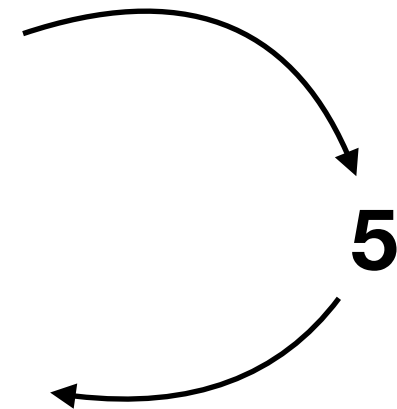
<https://fredfeng.github.io/Hunter/>

Hunter: Next-Generation Code Reuse for Java. FSE'16

A general synthesizer

```
func const: Int -> 0 | 1 | 2 | 3;  
func plus: Int -> Int, Int;  
func minus: Int -> Int, Int;  
func mult: Int r -> Int, Int;
```

```
0,  
plus(1,2)  
minus(3,1)  
plus(1,minus(3,1))  
...
```



Enumerate-check

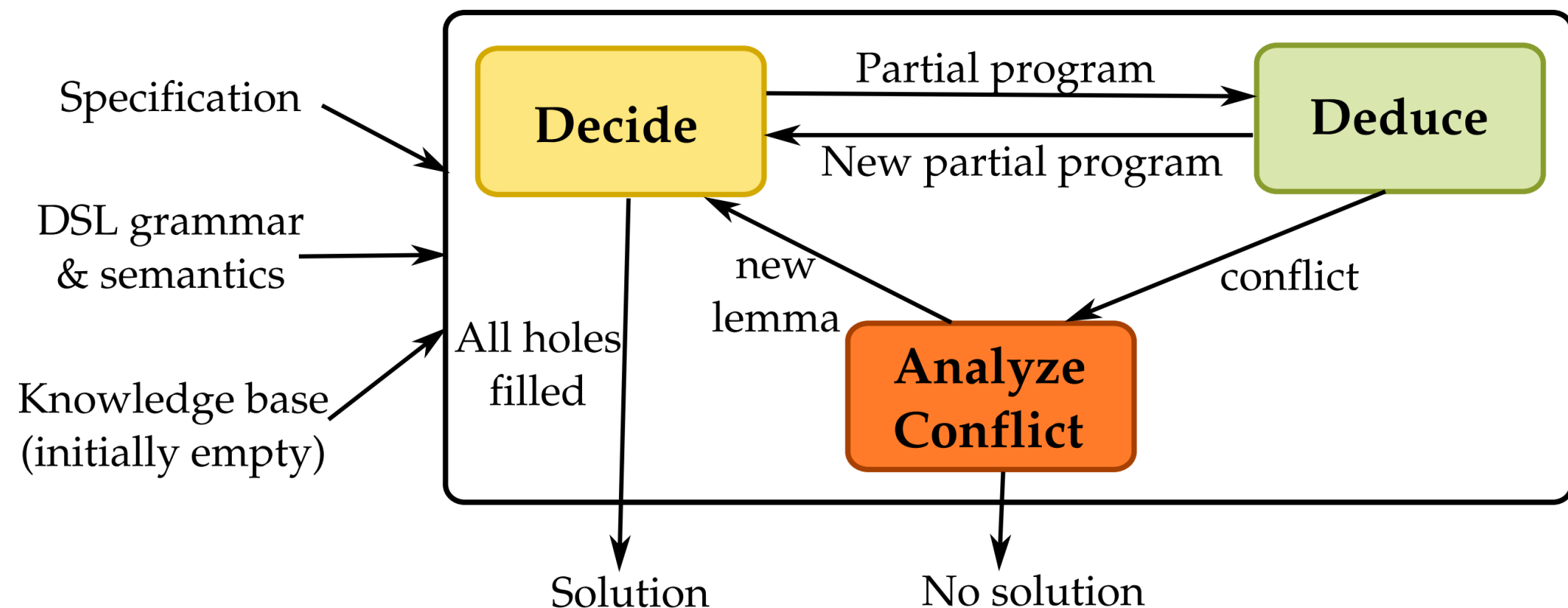


Prune invalid candidates (PL)



Enumerate promising candidates (ML)

Architecture of Neo framework



Neo in action: step 1



Declare
your DSL

First, specify the types that will be used

```
enum SmallInt {"0", "1", "2", "3"}
```

Finally, specify the production rules

```
func const: Int -> SmallInt;
```

```
func plus: Int -> Int, Int;
```

```
func minus: Int -> Int, Int;
```

```
func mult: Int -> Int, Int;
```

Neo in action: step 2

```
class ToyInterpreter(PostOrderInterpreter):  
  
    def eval_SmallInt(self, v):  
        return int(v)  
  
    def eval_const(self, node, args):  
        return args[0]  
  
    def eval_plus(self, node, args):  
        return args[0] + args[1]  
  
    def eval_minus(self, node, args):  
        return args[0] - args[1]  
  
    def eval_mult(self, node, args):  
        return args[0] * args[1]
```

Implement
the interpreter



Neo in action: step 3



Specify
your goal

```
synthesizer = Synthesizer(  
    enumerator=SmtEnumerator(spec, depth=3, loc=2),  
    decider=ExampleConstraintDecider(  
        spec=spec,  
        interpreter=ToyInterpreter(),  
        examples=[  
            Example(input=[4, 3], output=3),  
            Example(input=[6, 3], output=9),  
            Example(input=[1, 2], output=-2),  
            Example(input=[1, 1], output=0),  
        ]  
    )  
)
```

TODOs by next lecture

- Install Rosette and Neo
 - Install Rosette: https://docs.racket-lang.org/rosette-guide/ch_getting-started.html
 - Install Neo: <https://github.com/fredfeng/Trinity>
- Discuss your final project during the office hour!