Scallop and Neuro-Symbolic Programming

Lecture 4: Applications of Scallop and Conclusion

Agenda

1. **Case Study: Hand-Written Formula (HWF)**
   Scallop for Parsing and Evaluation

2. **Case Study: CLEVR**
   Scallop for Interpreting CLEVR Program

3. **Case Study: CLUTRR**
   Scallop for Transitivity Reasoning

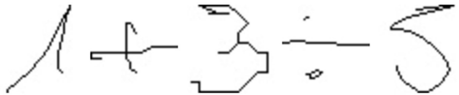
4. **Conclusion**
   Future Works and

Case Study: Hand-Written Formula

Hand written formula parsing and evaluation

Input

Output



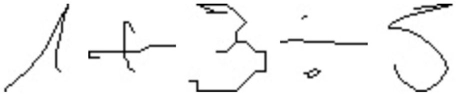
1.6

(Assuming Image Segmentation is Done)
Input Data is a sequence of symbol images

Hand written formula parsing and evaluation

Input

Output



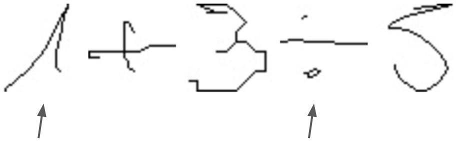
1.6

Each symbol can be one from 14 symbols: 0-9, +, -, *, /

Hand written formula parsing and evaluation

Input

Output



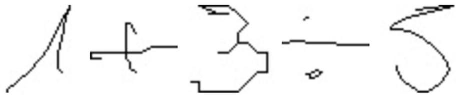
```
0.01::symbol(0, “0”)      0.01::symbol(3, “0”)
0.80::symbol(0, “1”)      0.01::symbol(3, “1”)
0.01::symbol(0, “2”)      0.01::symbol(3, “2”)
0.01::symbol(0, “3”)      0.01::symbol(3, “3”)
0.01::symbol(0, “4”)      0.01::symbol(3, “4”)
0.01::symbol(0, “5”)      0.01::symbol(3, “5”)
0.01::symbol(0, “6”)      0.01::symbol(3, “6”)
0.01::symbol(0, “7”)      0.01::symbol(3, “7”)
0.01::symbol(0, “8”)      0.01::symbol(3, “8”)
0.01::symbol(0, “9”)      0.01::symbol(3, “9”)
0.01::symbol(0, “+”)      0.01::symbol(3, “+”)
0.01::symbol(0, “-”)      0.01::symbol(3, “-”)
0.01::symbol(0, “*”)      0.01::symbol(3, “*”)
0.01::symbol(0, “/”)      0.78::symbol(3, “/”)
```

1.6

Hand written formula parsing and evaluation

Input

Output



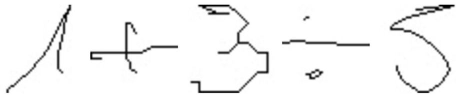
1.6

```
# a Context-Free Grammar for Arith
Expression
<P> ::= <S> # the start rule
<S> ::= <S> "+" <M> | <S> "-" <M> | <M>
<M> ::= <M> "*" <T> | <M> "/" <T> | <T>
<T> ::= 0 | 1 | … | 9
```

Hand written formula parsing and evaluation

Input

Output



1.6

```
# a Context-Free Grammar for Arith
Expression
<P> ::= <S> # the start rule
<S> ::= <S> "+" <M> | <S> "-" <M> | <M>
<M> ::= <M> "*" <T> | <M> "/" <T> | <T>
<T> ::= 0 | 1 | … | 9
```

The ”*”/”/” symbol has a higher precedence than “+”/”-”

Hand written formula parsing and evaluation

```
# a Context-Free Grammar for Arith Expression
<P> ::= <S> # the start rule
<S> ::= <S> "+" <M> | <S> "-" <M> | <M>
<M> ::= <M> "*" <T> | <M> "/" <T> | <T>
<T> ::= 0 | 1 | … | 9
```

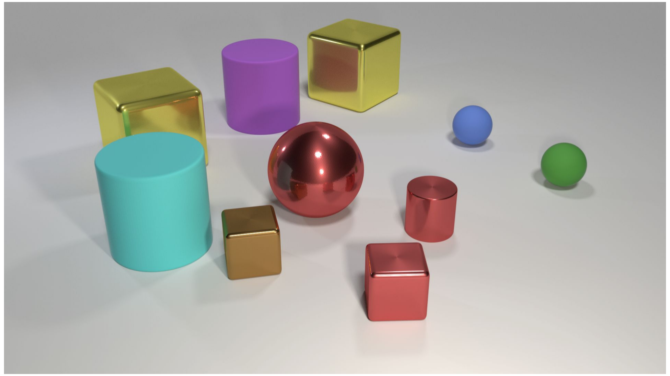
```
rel value_node(x, v) =
    symbol(x, d), digit(d, v), length(n), x < n
rel mult_div_node(x, "v", x, x, x, x, x) =
    value_node(x, _)
rel mult_div_node(h, s, x, l, end, begin, end) =
    symbol(x, s), mult_div(s), node_id_hash(x, s, l, end, h),
    mult_div_node(l, _, _, _, _, begin, x - 1),
    value_node(end, _), end == x + 1
rel plus_minus_node(x, t, i, l, r, begin, end) =
    mult_div_node(x, t, i, l, r, begin, end)
rel plus_minus_node(h, s, x, l, r, begin, end) =
    symbol(x, s), plus_minus(s), node_id_hash(x, s, l, r, h),
    plus_minus_node(l, _, _, _, _, begin, x - 1),
    mult_div_node(r, _, _, _, _, x + 1, end)
```

Hand written formula parsing and evaluation

```
rel eval(x, y, x, x) = value_node(x, y)
rel eval(x, y1 + y2, b, e) = plus_minus_node(x, "+", i, l, r, b, e), eval(l, y1, b, i - 1), eval(r, y2, i + 1, e)
rel eval(x, y1 - y2, b, e) = plus_minus_node(x, "-", i, l, r, b, e), eval(l, y1, b, i - 1), eval(r, y2, i + 1, e)
rel eval(x, y1 * y2, b, e) = mult_div_node(x, "*", i, l, r, b, e), eval(l, y1, b, i - 1), eval(r, y2, i + 1, e)
rel eval(x, y1 / y2, b, e) = mult_div_node(x, "/", i, l, r, b, e), eval(l, y1, b, i - 1), eval(r, y2, i + 1, e), y2 != 0.0
```

Case Study: CLEVR

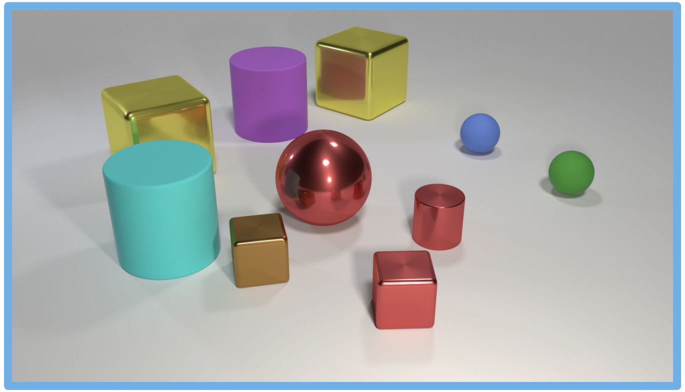
CLEVR: Visual and Spatial Reasoning



Q: Are there an equal number of large things and metal spheres?

A: No (False)

CLEVR: Visual and Spatial Reasoning

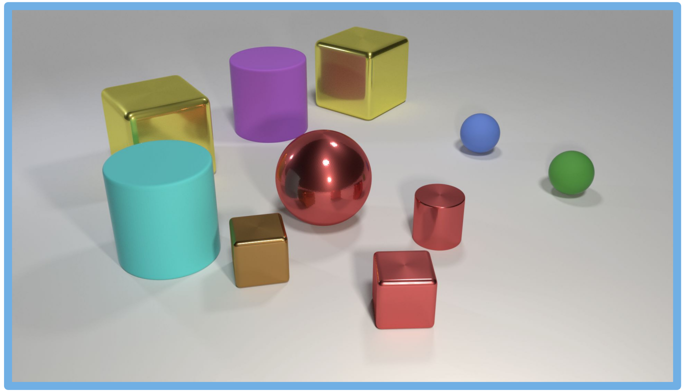


Q: Are there an equal number of large things and metal spheres?

A: No (False)

Input

CLEVR: Visual and Spatial Reasoning

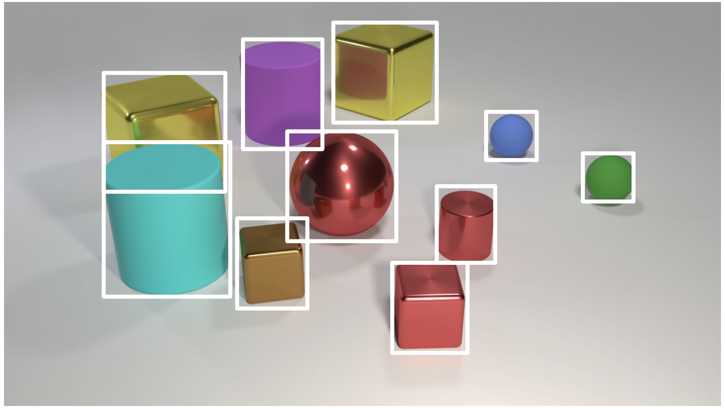


Q: Are there an equal number of large things and metal spheres?

A: No (False)

Output

CLEVR: Visual and Spatial Reasoning

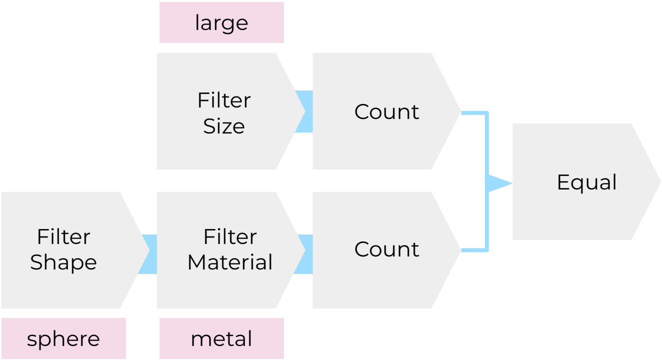


```
0.02::obj_color(1, "red")
0.94::obj_color(1, "green")
0.02::obj_color(1, "blue")
0.02::obj_color(1, "yellow")
0.03::obj_shape(1, "cube")
0.03::obj_shape(1, "cylinder")
0.94::obj_shape(1, "sphere")

0.06::obj_size(1, "big")
0.94::obj_size(1, "small")

0.06::obj_material(1, "metal")
0.94::obj_material(1, "rubber")

0.02::left(1, 2)
0.04::behind(1, 2)
0.01::left(1, 3)
0.87::behind(1, 3)
…
```

Symbolic Scene Graph

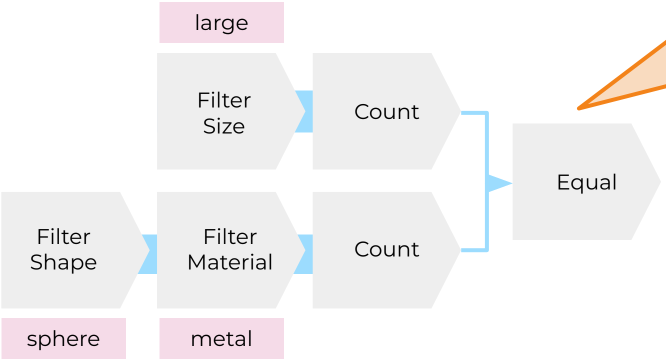
CLEVR: Visual and Spatial Reasoning

Q: Are there an equal number of large things and metal spheres?



CLEVR: Visual and Spatial Reasoning

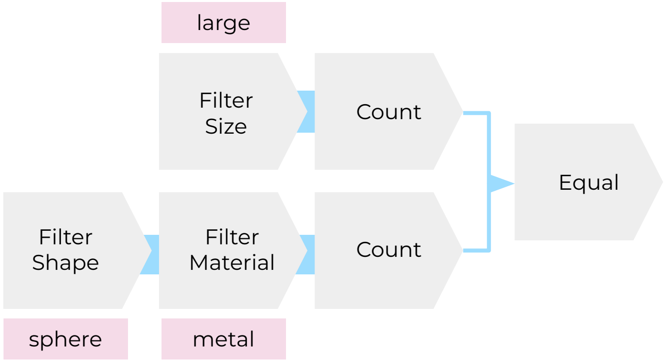
Q: Are there an equal number of large things and metal spheres?



Natural Language Question represented in programmatic query

CLEVR: Visual and Spatial Reasoning

Q: Are there an equal number of large things and metal spheres?

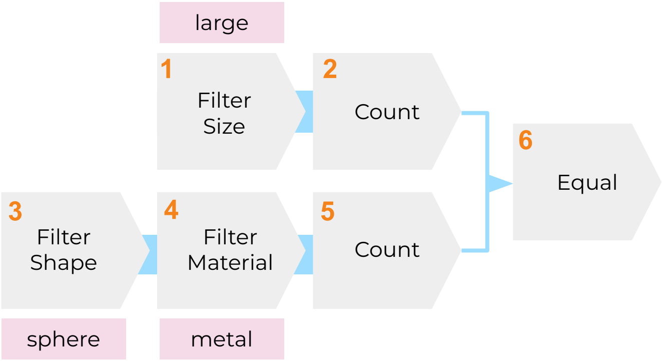


```
type filter_size_expr(usize, usize, String)
type filter_material_expr(usize, usize, String)
type filter_shape_expr(usize, usize, String)
type equal_expr(usize, usize, usize)
type count_expr(usize, usize)
type scene_expr(usize)
```

expression ID,
Input expression ID,
argument

CLEVR: Visual and Spatial Reasoning

Q: Are there an equal number of large things and metal spheres?



```
type filter_size_expr(usize, usize, String)
type filter_material_expr(usize, usize, String)
type filter_shape_expr(usize, usize, String)
type equal_expr(usize, usize, usize)
type count_expr(usize, usize)
type scene_expr(usize)
```

```
scene_expr(0)
filter_size_expr(1, 0, "large")
count_expr(2, 1)
filter_shape_expr(3, 0, "sphere")
filter_material_expr(4, 3, "metal")
count_expr(5, 4)
equal_expr(6, 2, 5)
```

CLEVR: Visual and Spatial Reasoning

```
0.02::obj_color(1, "red")
0.94::obj_color(1, "green")
0.02::obj_color(1, "blue")
0.02::obj_color(1, "yellow")
…
```

Symbolic Scene Graph

```
scene_expr(0)
filter_size_expr(1, 0, "large")
count_expr(2, 1)
filter_shape_expr(3, 0, "sphere")
…
```

Symbolic Programmatic Query

```
rel eval_objs(e, o) = filter_size_expr(e, f, s),
                      eval_objs(f, o), obj_size(o, s)
rel eval_objs(e, o) = filter_material_expr(e, f, m),
                      eval_objs(f, o), obj_material(o, m)
rel eval_objs(e, o) = filter_shape_expr(e, f, s),
                      eval_objs(f, o), obj_shape(o, s)
rel eval_num(e, n) = n = count(o: eval_objs(f, o) where e: count_expr(e, f))
rel eval_yn(e, x == y) = equal_expr(e, a, b), eval_num(a, x), eval_num(b, y)
```

Query Interpreter & Evaluator

CLEVR: Visual and Spatial Reasoning

```
0.02::obj_color(1, "red")
0.94::obj_color(1, "green")
0.02::obj_color(1, "blue")
0.02::obj_color(1, "yellow")
…
```

Symbolic Scene Graph

```
scene_expr(0)
filter_size_expr(1, 0, "large")
count_expr(2, 1)
filter_shape_expr(3, 0, "sphere")
…
```

Symbolic Programmatic Query

```
rel eval_objs(e, o) = filter_size_expr(e, f, s),
                      eval_objs(f, o), obj_size(o, s)
rel eval_objs(e, o) = filter_material_expr(e, f, m),
                       eval_objs(f, o), obj_material(o, m)
rel eval_objs(e, o) = filter_shape_expr(e, f, s),
                      eval_objs(f, o), obj_shape(o, s)
rel eval_num(e, n) = n = count(o: eval_objs(f, o) where e: count_expr(e, f))
rel eval_yn(e, x == y) = equal_expr(e, a, b), eval_num(a, x), eval_num(b, y)
```

Query Interpreter & Evaluator

Query Output

```
0.002::(True)
0.998::(False)
```

Case Study: CLUTRR

CLUTRR: Kinship Reasoning

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.

Question:

What is the relationship between **Ruth** and **Sheila**?

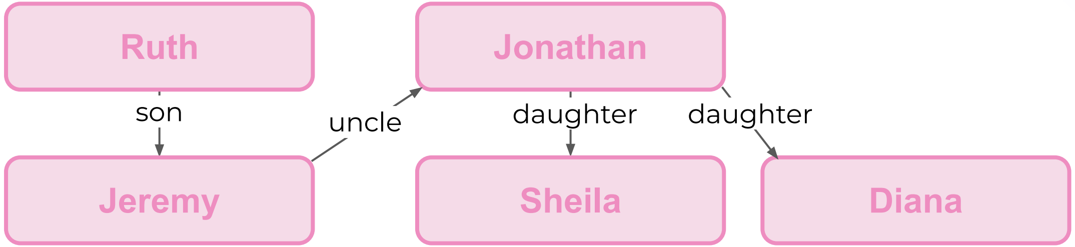
CLUTRR: Kinship Reasoning

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.

Question:

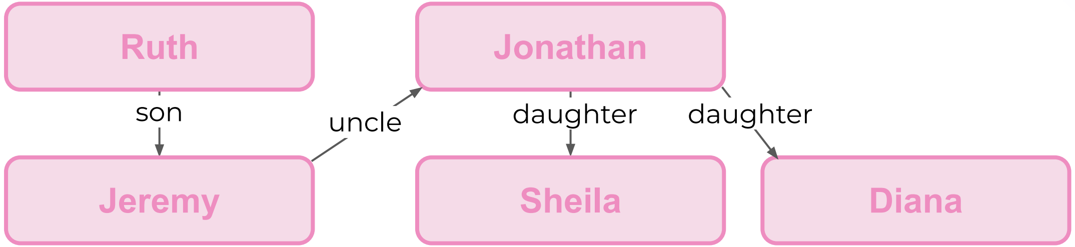
What is the relationship between **Ruth** and **Sheila**?



CLUTRR: Kinship Reasoning

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.



Question:

What is the relationship between **Ruth** and **Sheila**?

Answer:

Sheila is **Ruth**'s niece.

CLUTRR: Kinship Reasoning

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana].

However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.

```
0.951::context(“daughter”, “Cristina”, “Sheila”)
0.002::context(“mother”, “Cristina”, “Sheila”)
0.004::context(“father”, “Cristina”, “Sheila”)
…
0.001::context(“no_rela”, “Cristina”, “Sheila”)
0.942::context(“daughter”, “Christina”, “Diana”)
0.015::context(“mother”, “Christina”, “Diana”)
…
0.002::context(“no_rela”, “Sheila”, “Diana”)
```

Symbolic Context

For each pair of names, classify them into 21 types of kinship

CLUTRR: Kinship Reasoning

```
type transitive(r1: String, r2: String, r3: String)
rel transitive("son", "uncle", "brother")
rel transitive("brother", "daughter", "niece")
rel derived(r, s, o) = context(r, s, o)
rel derived(r3, x, z) = transitive(r1, r2, r3),
                        derived(r1, x, y),
                        derived(r2, y, z), x != z
```

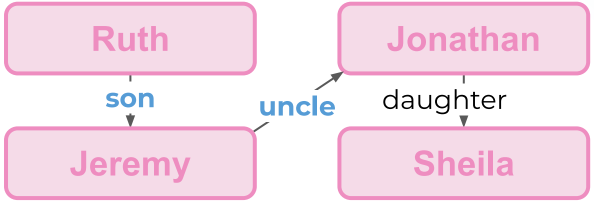
CLUTRR: Kinship Reasoning

```
type transitive(r1: String, r2: String, r3: String)
rel transitive("son", "uncle", "brother")
rel transitive("brother", "daughter", "niece")
rel derived(r, s, o) = context(r, s, o)
rel derived(r3, x, z) = transitive(r1, r2, r3),
                        derived(r1, x, y),
```

Define high-order relationship with transitivity rule

CLUTRR: Kinship Reasoning

```
type transitive(r1: String, r2: String, r3: String)
rel transitive("son", "uncle", "brother")
rel transitive("brother", "daughter", "niece")
rel derived(r, s, o) = context(r, s, o)
rel derived(r3, x, z) = transitive(r1, r2, r3),
                        derived(r1, x, y),
                        derived(r2, y, z), x != z
```



Ruth

son

Jeremy

uncle

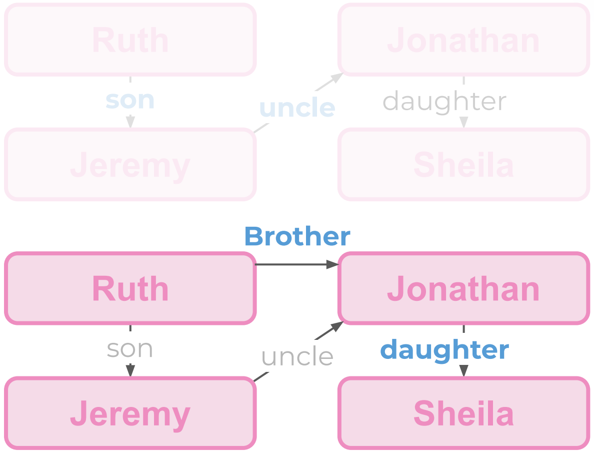
Jonathan

daughter

Sheila

CLUTRR: Kinship Reasoning

```
type transitive(r1: String, r2: String, r3: String)
rel transitive("son", "uncle", "brother")
rel transitive("brother", "daughter", "niece")
rel derived(r, s, o) = context(r, s, o)
rel derived(r3, x, z) = transitive(r1, r2, r3),
                        derived(r1, x, y),
                        derived(r2, y, z), x != z
```

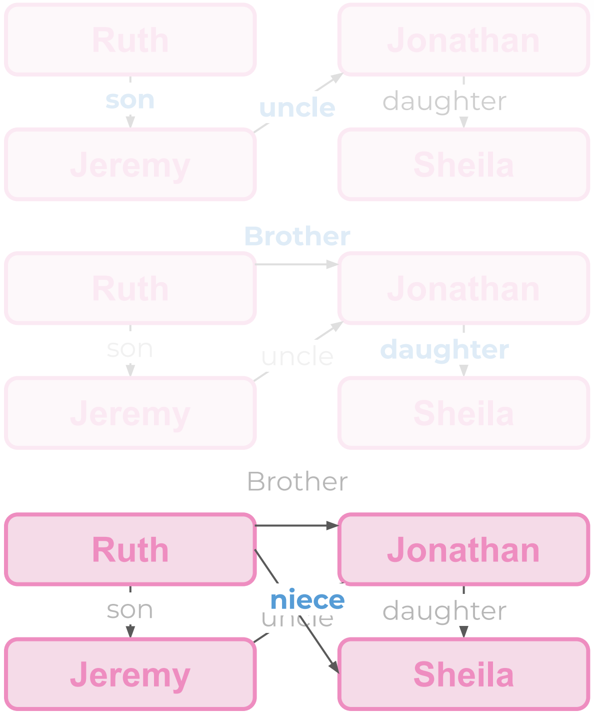


CLUTRR: Kinship Reasoning

```
type transitive(r1: String, r2: String, r3: String)
rel transitive("son", "uncle", "brother")
rel transitive("brother", "daughter", "niece")
rel derived(r, s, o) = context(r, s, o)
rel derived(r3, x, z) = transitive(r1, r2, r3),
                        derived(r1, x, y),
                        derived(r2, y, z), x != z
```

Answer:

Sheila is **Ruth**’s niece.



Conclusion and Future Works

What we have learned

- Why is there a need for Neuro-Symbolic method
- How differentiable & probabilistic reasoning can bridge the gap between perception and symbols
- Scallop, a programming language based on Datalog
- Writing probabilistic programs in Scallop
- The concept of tag and provenance and how that facilitates differentiable and probabilistic reasoning
- Learn to integrate Scallop with machine learning frameworks like PyTorch, and actually run the machine learning experiments
- A few more applications involving perception and reasoning

Where will Scallop go

- Let applications drive the core development of Scallop!
- Programming Languages
    - Program Analysis, Bug Finding, Formal Verification
- Computer Vision (CV)
    - Visual Question Answering, 3D Scene Reasoning
- Natural Language Processing (NLP)
    - Procedural Reasoning and Events
- Reinforcement Learning and Game Playing
- Temporal Logic and Time Series Reasoning

Thank you!