Oruga

Implementation and Use of Representational Systems Theory

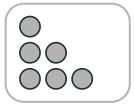
Daniel Raggi Gem Stapleton Aaron Stockdill
Grecia Garcia Garcia Peter C.-H. Cheng Mateja Jamnik

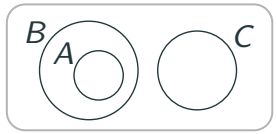
University of Cambridge, UK
University of Sussex, UK

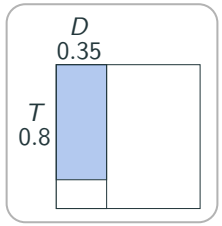
What are representations?

What are representations?

$1 + 2 + 3$



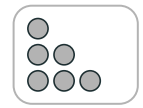


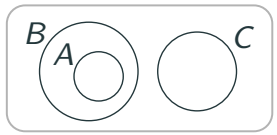


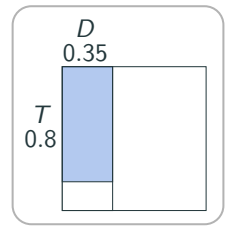
What are representations?

How can we talk about their structure?

$1 + 2 + 3$

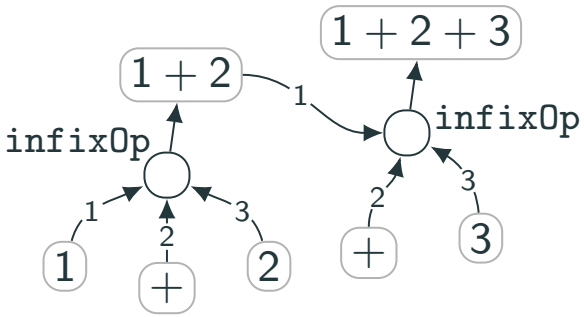


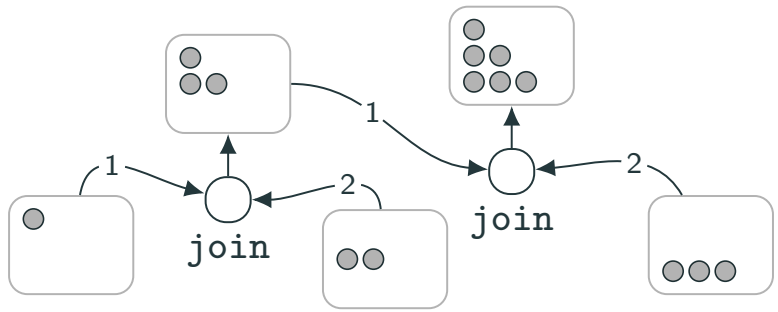


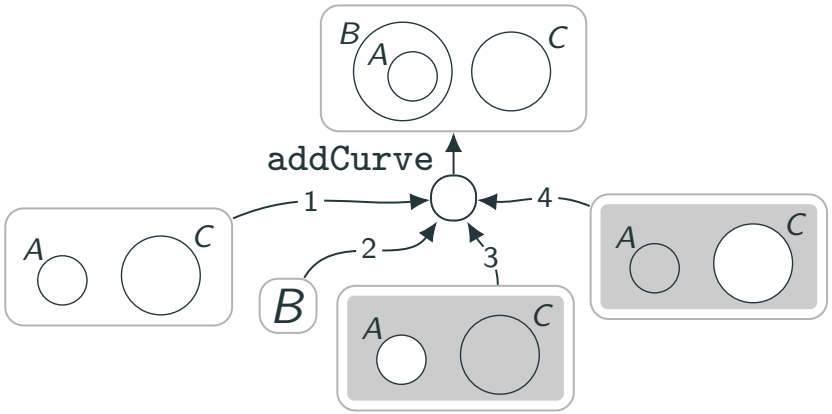


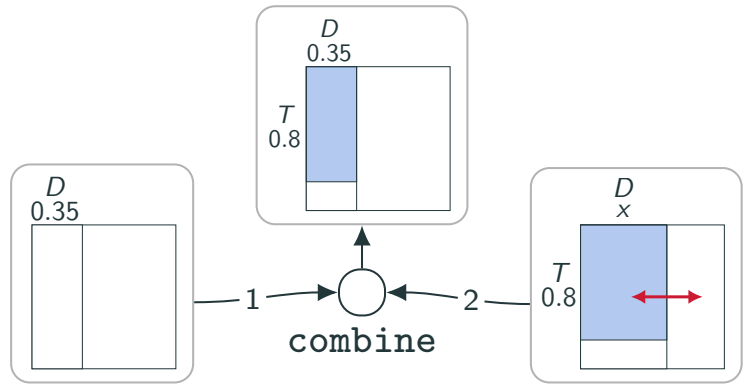
What are representations?

How can we talk about their structure?



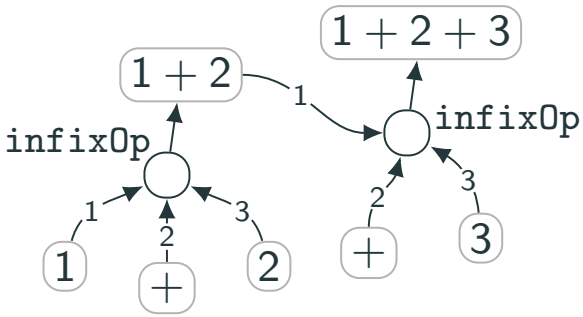


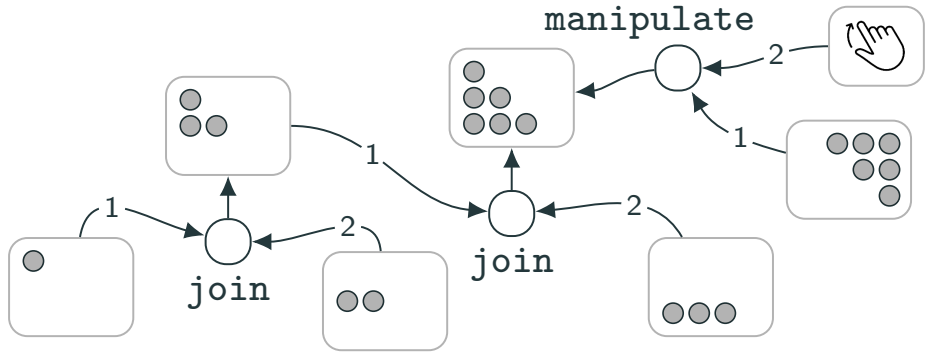


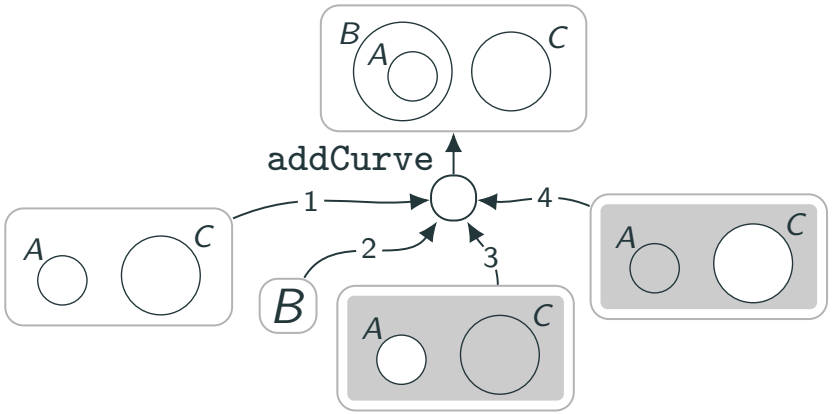


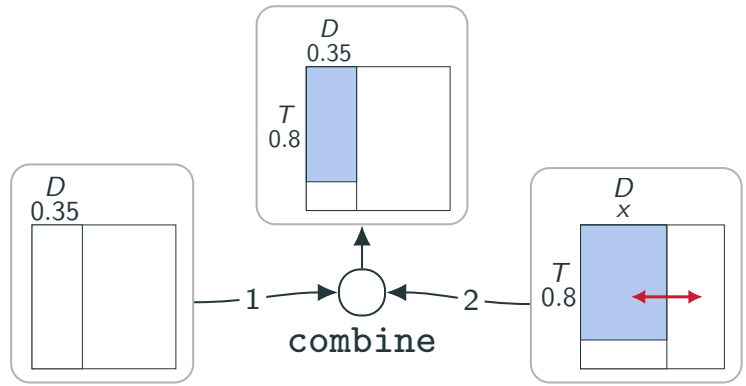
What are representations?

How can we talk about their structure?



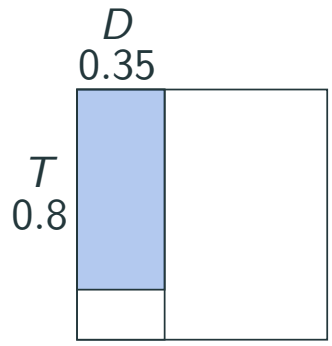






Can we use the relations between representations across systems?

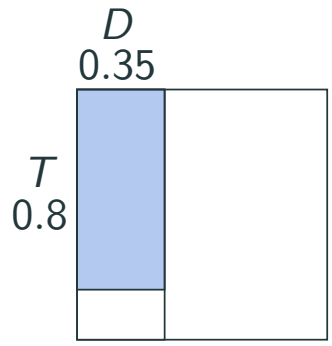
$$\Pr(D) = 0.35,$$
$$\Pr(T|D) = 0.8$$



Can we use the relations between representations across systems?

Are there methods to transform representations across systems?

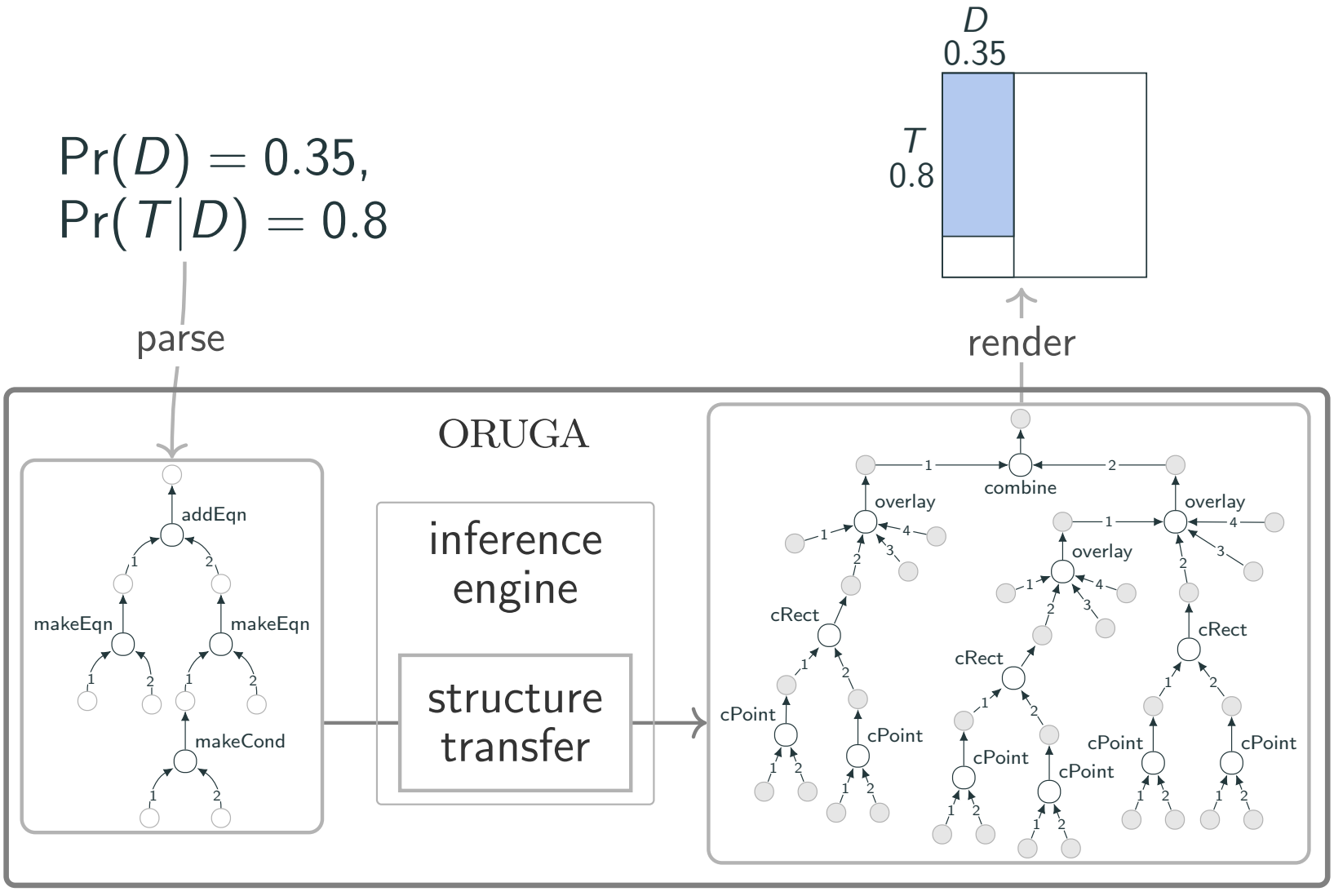
$$\Pr(D) = 0.35,$$
$$\Pr(T|D) = 0.8$$



Can we use the relations between representations across systems?

Are there methods to transform representations across systems?

$$\Pr(D) = 0.35,$$
$$\Pr(T|D) = 0.8$$



Using Oruga

Specifying grammars in Oruga

Type Systems

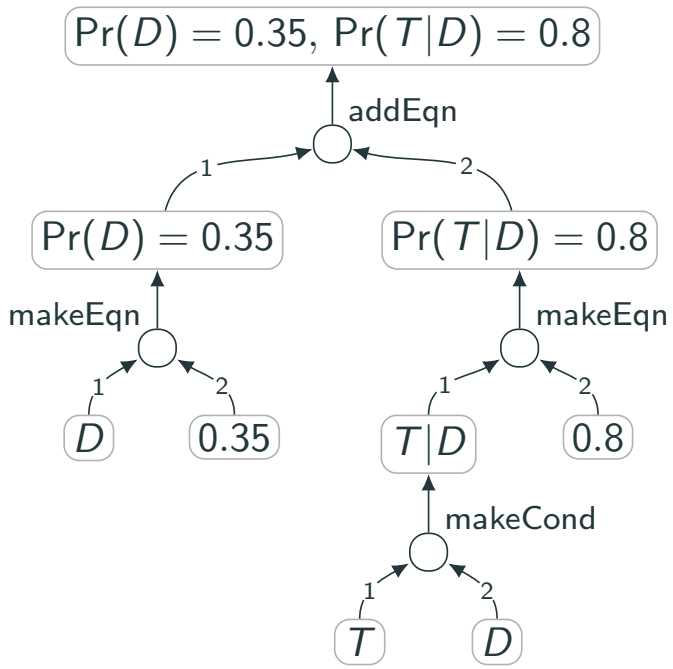
```
typeSystem bayes =
  imports {arith, eventT}
  types
    {
      _:probEqn, _:probSys,
      _:condEvent, _:genEvent,
      inter, union, binOp
    }
  order
    {
      probEqn < probSys,
      inter < binOp, union < binOp,
      event < genEvent,
      condEvent < genEvent
    }
```

Constructor Specifications

```
conSpec bayesG:bayes =
  imports {arithG, event}
  constructors
    {
      makeEqn : [genEvent,numExp] -> probEqn,
      addEqn : [probEqn,probSys] -> probSys,
      makeCond : [event,event] -> condEvent,
      complement : [event] -> event,
      infixOp : [event,bin,event] -> event
    }
```

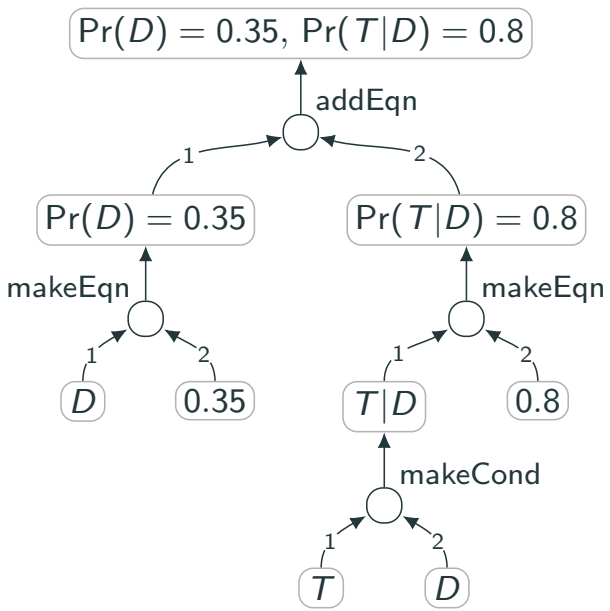
Graphs in Oruga

```
graph g:bayesG = {t:tt:probSys
                    <- addEqn[t1:tt1:probEqn
                                <- makeEqn[t11:D:event,
                                           t12:0.35:real10],
                              t2:tt2:probEqn
                                <- makeEqn[t21:T|D:condEvent
                                             <- makeCond[t211:T:event,
                                                         t212:D:event],
                                           t22:0.8:real10]]}
```



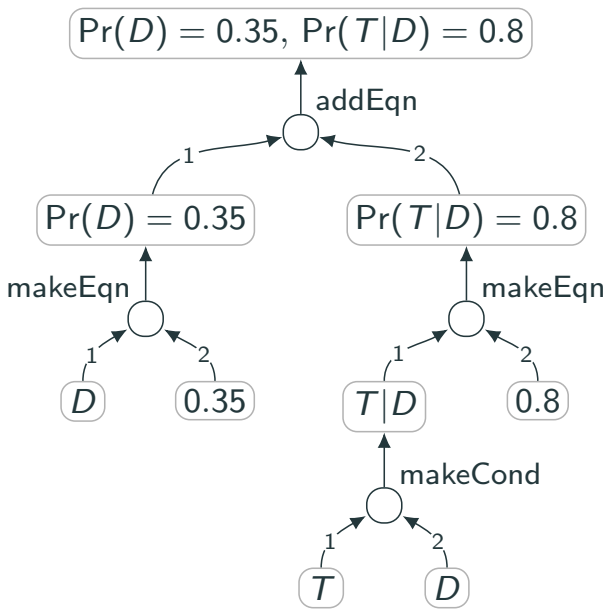
Transformations with Oruga using schemas

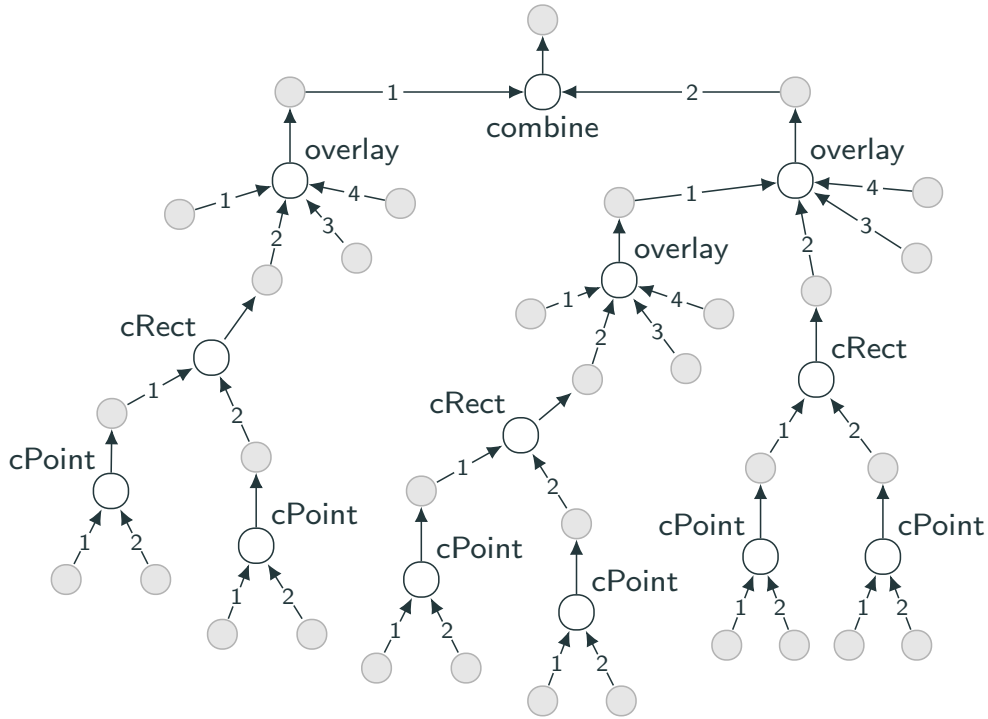
```
transfer
  sourceGraph g
  targetConSpec areaDiagramG
  interConSpec interBayesArea
  goalGraph {:metaTrue <- encode[t:tt:probSys,t':area]}
  outputLimit 10
  searchLimit 400
  output bayesArea
```



Transformations with Oruga using schemas

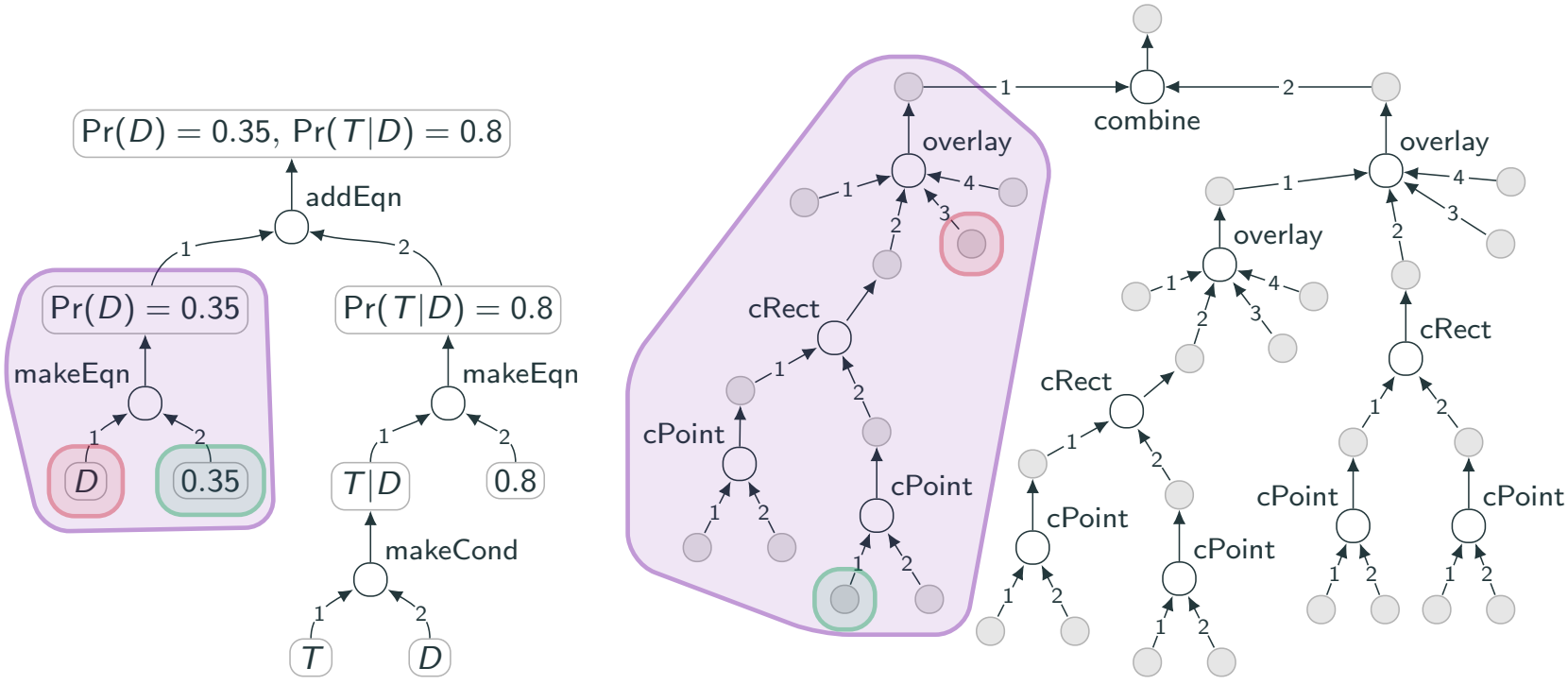
```
transfer
  sourceGraph g
  targetConSpec areaDiagramG
  interConSpec interBayesArea
  goalGraph {:metaTrue <- encode[t:tt:probSys,t':area]}
  outputLimit 10
  searchLimit 400
  output bayesArea
```



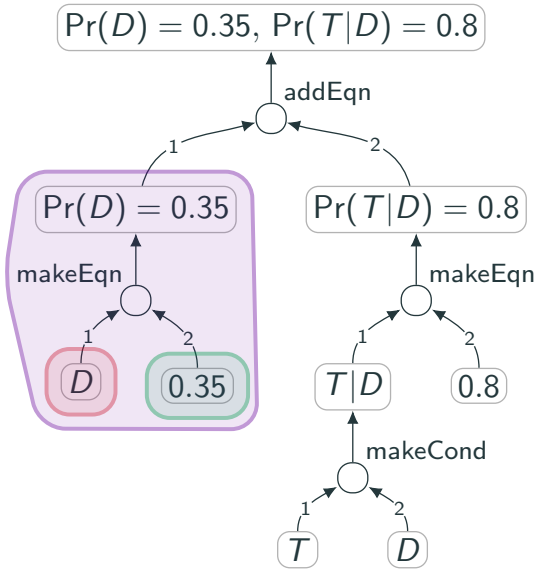


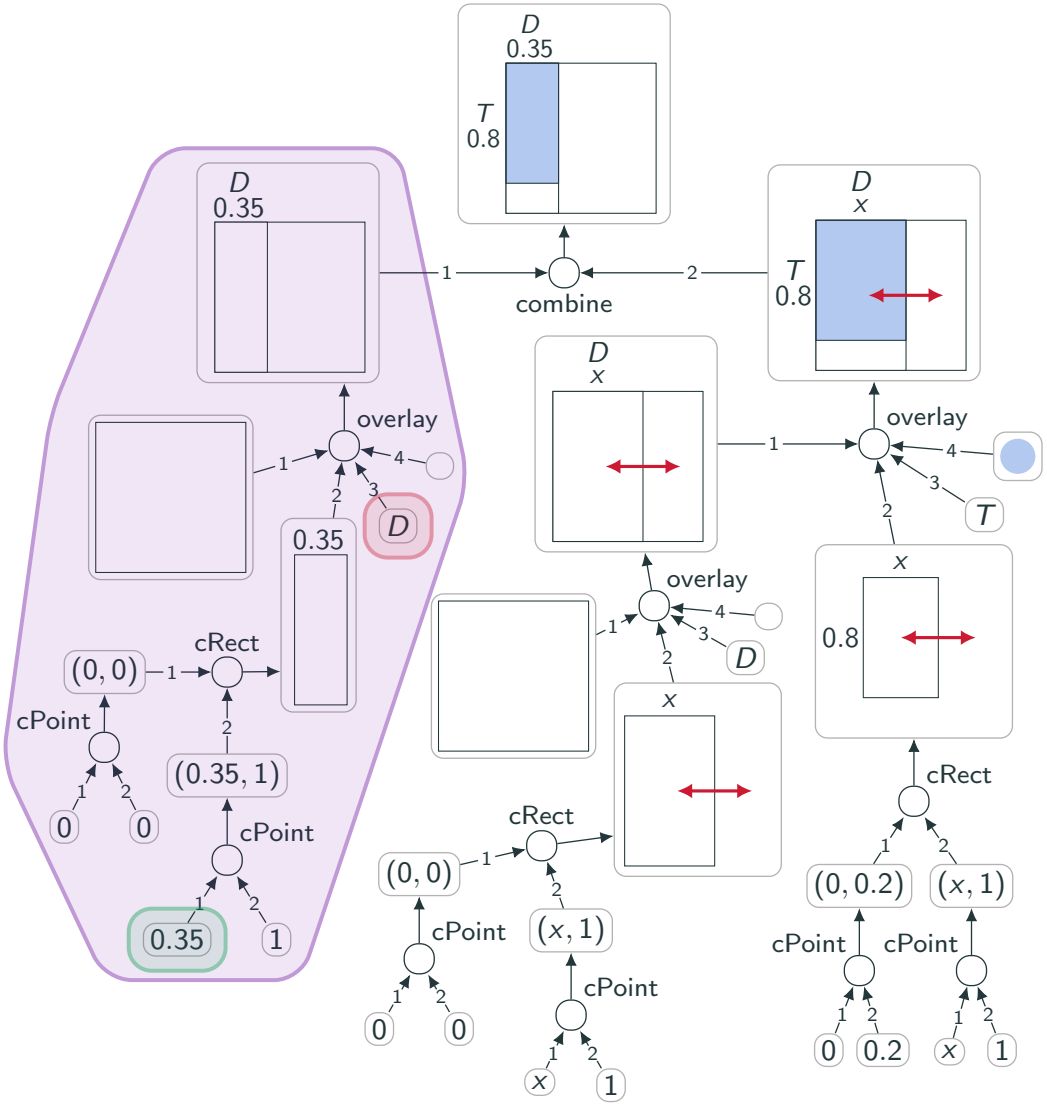
Transformations with Oruga using schemas

```
transfer
  sourceGraph g
  targetConSpec areaDiagramG
  interConSpec interBayesArea
  goalGraph {:metaTrue <- encode[t:tt:probSys,t':area]}
  outputLimit 10
  searchLimit 400
  output bayesArea
```



Transformations with Oruga using schemas





Demo

.oruga document $\rightarrow$ run $\rightarrow$ .tex file

Additional material

Schemas

```
tSchema constructEvent:(bayesG,areaDiagramG,interBayesArea) =
  source {t:probEqn <- makeEqn[t1:events,t2:numExp]}
  target {t':area
          <- overlayRect[t1':empty,
                         t2':rect
                             <- cRect[t21':point
                                      <- cPoint[t211':0:real10,
                                                t212':0:real10],
                                      t22':point
                                      <- cPoint[t221':numExp,
                                                t222':1:real10]],
                         t3':tag,
                         t4':blue]}
  antecedent {:true <- match[t1:events,t3':tag],
              :true <- equal[t2:numExp,t221':numExp]}
  consequent {:true <- encode[t:probEqn,t':area]}
  strength 1
```

"the probability of an event is analogous to the area of a region"

Arithmetic and Dot Diagrams