An Empirical Study of Partial Deduction for miniKanren

Kate Verbitskaia, Daniil Berezun, Dmitry Boulytchev

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg State University

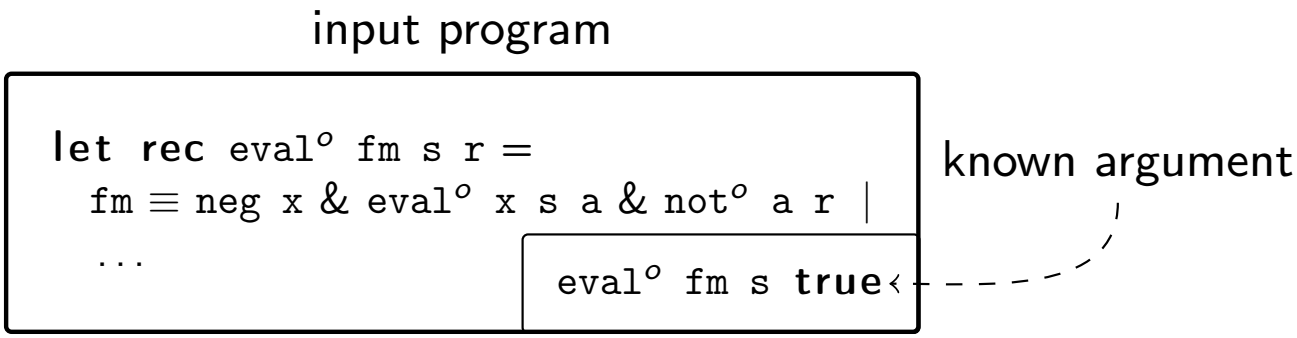
27.08.2020

Specialization: a Method to Improve Programs

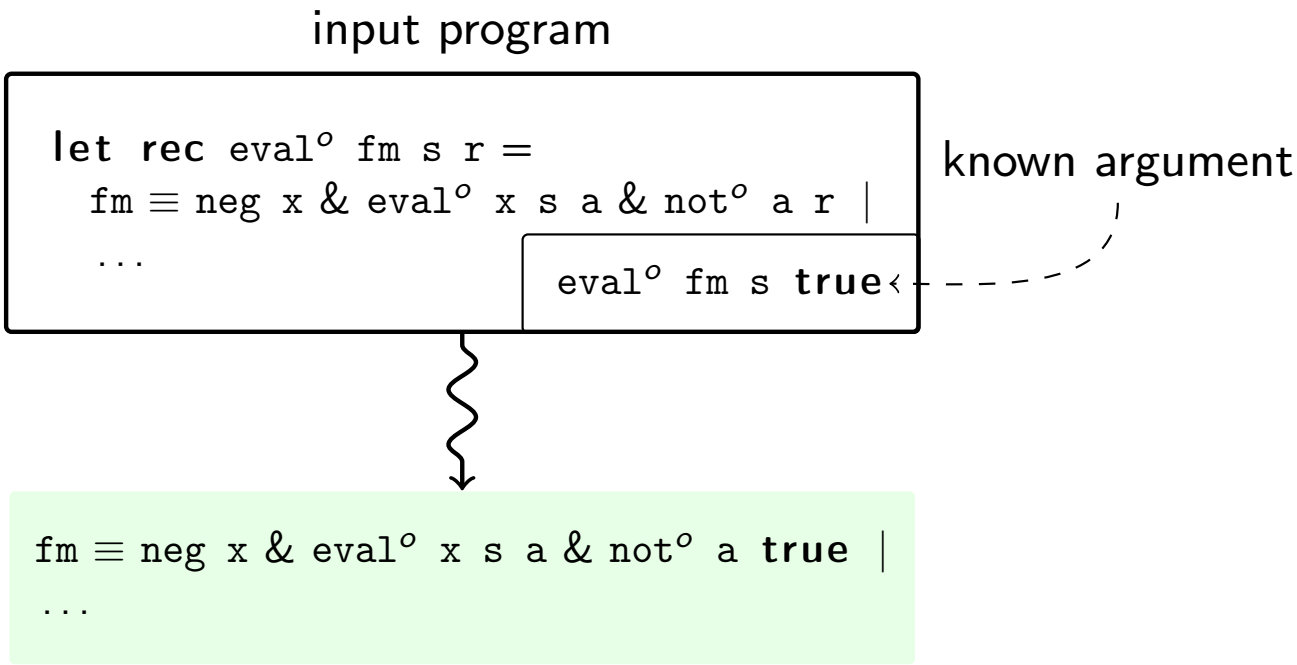
input program

```
let rec eval^o fm s r =
  fm ≡ neg x & eval^o x s a & not^o a r |
  ...
```

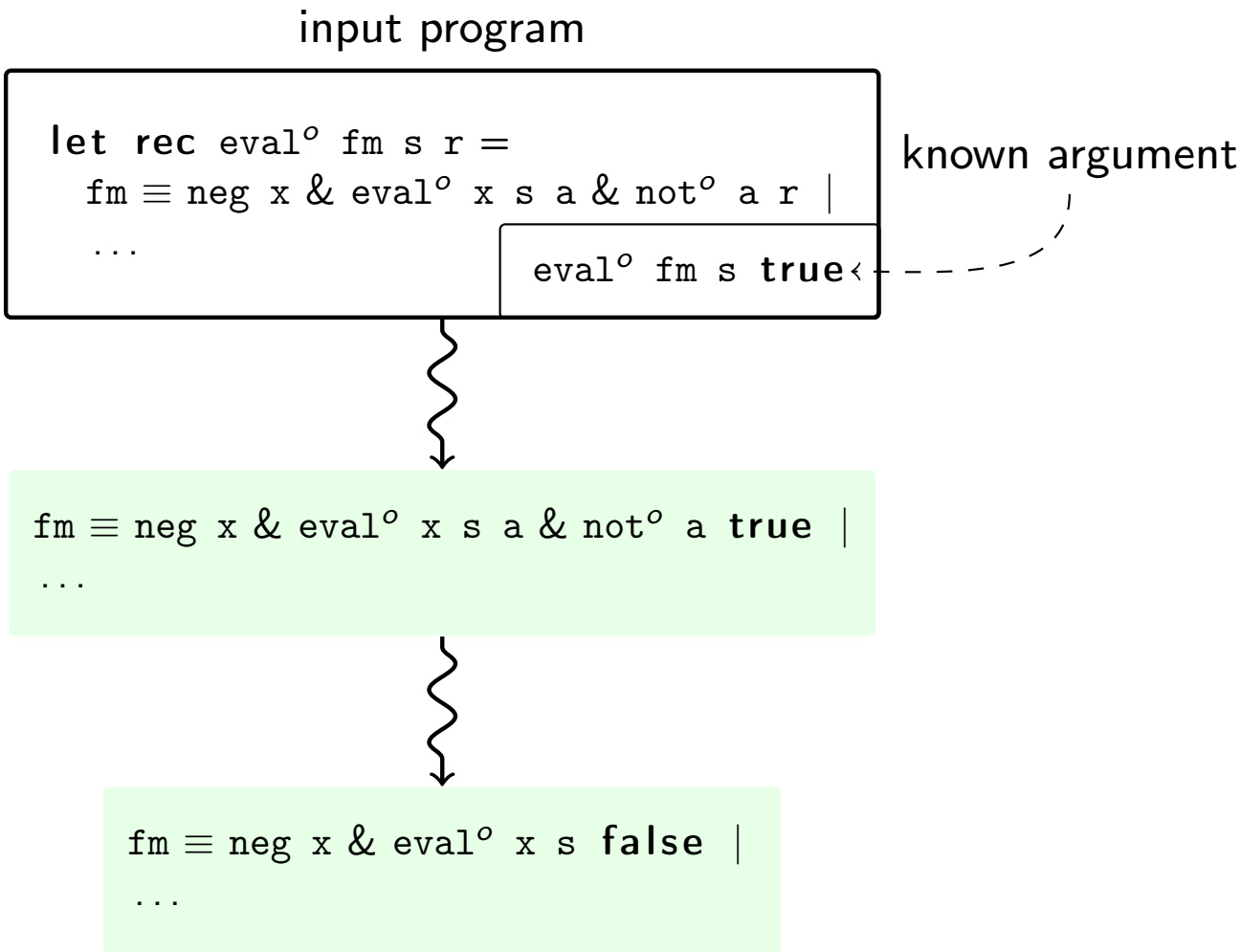
Specialization: a Method to Improve Programs



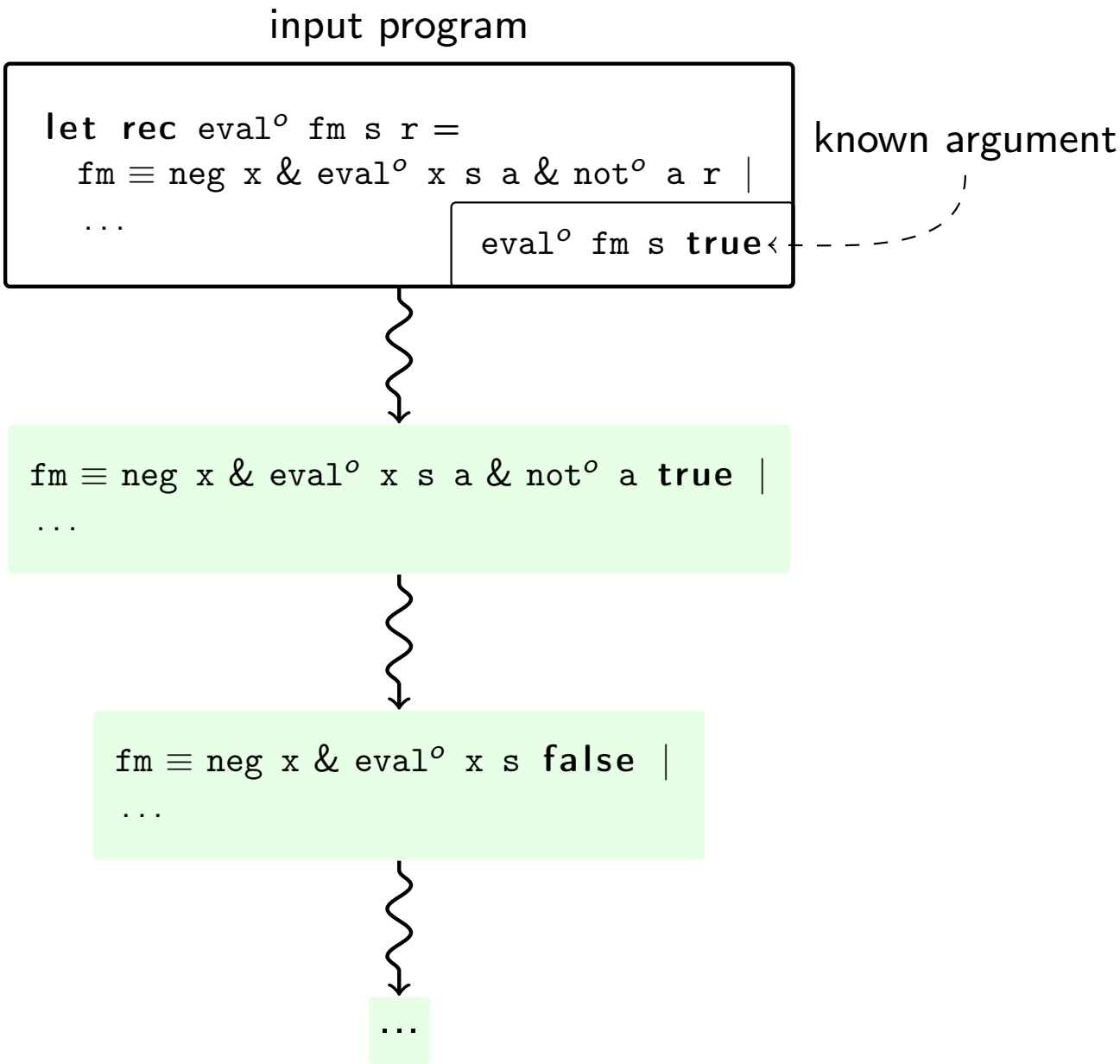
Specialization: a Method to Improve Programs



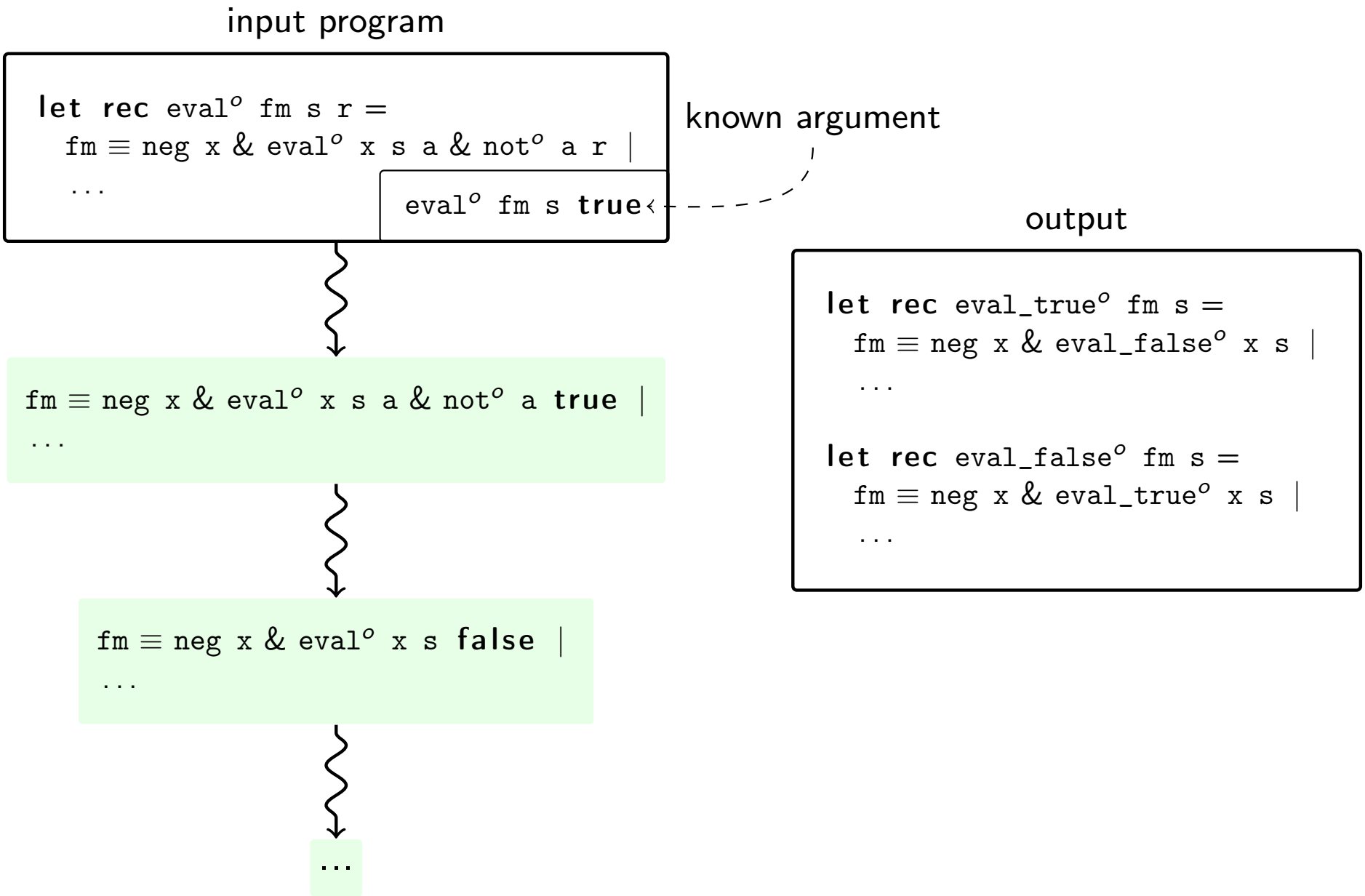
Specialization: a Method to Improve Programs



Specialization: a Method to Improve Programs



Specialization: a Method to Improve Programs



Partial Deduction: Specialization for Logic Programming

input

```
let double_appendᵒ x y z r =
  ocanren {
    fresh t in
      appendᵒ x y t &
      appendᵒ t z r}

let rec appendᵒ x y r =
  ocanren {
    (x ≡ [] & y ≡ r) |
    (fresh h x' r' in
      x ≡ h :: x' &
      appendᵒ x' y r' &
      r ≡ h :: r')}
```

```
double_appendᵒ x y z r
```

output

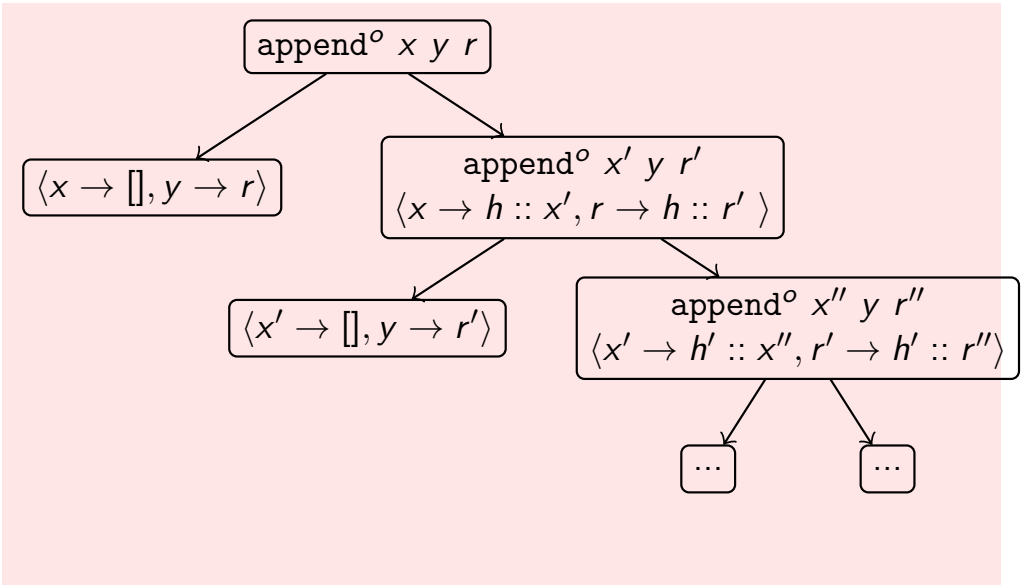
```
let double_appendᵒ x y z r =
  ocanren {
    (x ≡ [] & appendᵒ y z r) |
    (fresh h x' r' in
      x ≡ h :: x' &
      double_appendᵒ x' y z r' &
      r ≡ h :: r')}

let rec appendᵒ x y r =
  ocanren {
    (x ≡ [] & y ≡ r) |
    (fresh h x' r' in
      x ≡ h :: x' &
      appendᵒ x' y r' &
      r ≡ h :: r')}
```

Partial Deduction for miniKanren: Bird's-eye View

input

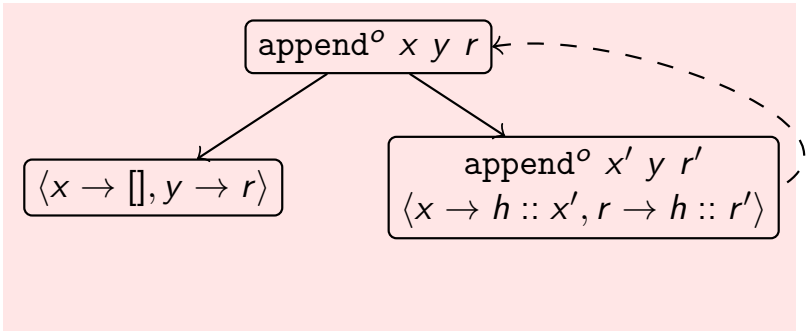
```
let rec append^o x y r =
  ocanren {
    (x ≡ [] & y ≡ r) |
    (fresh h x' r' in
      x ≡ h :: x' &
      append^o x' y r' &
      r ≡ h :: r')}
```

```
append^o x y r
```



output

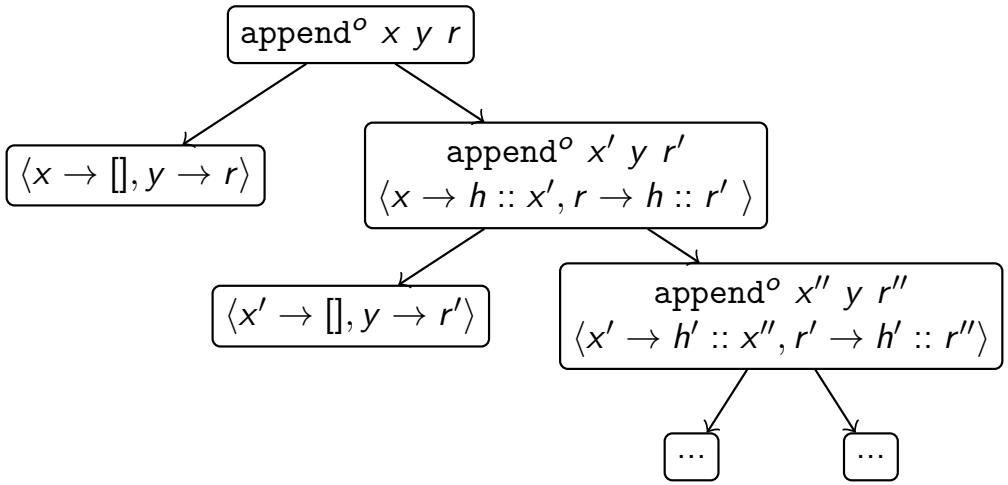
```
let rec append^o x y r =
  ocanren { fresh h t r' in
    (x ≡ [] & y ≡ r) |
    ( x ≡ h :: x' &
      append^o x' y r' &
      t ≡ h :: r')}
```



Partial Deduction: Bird's-eye View

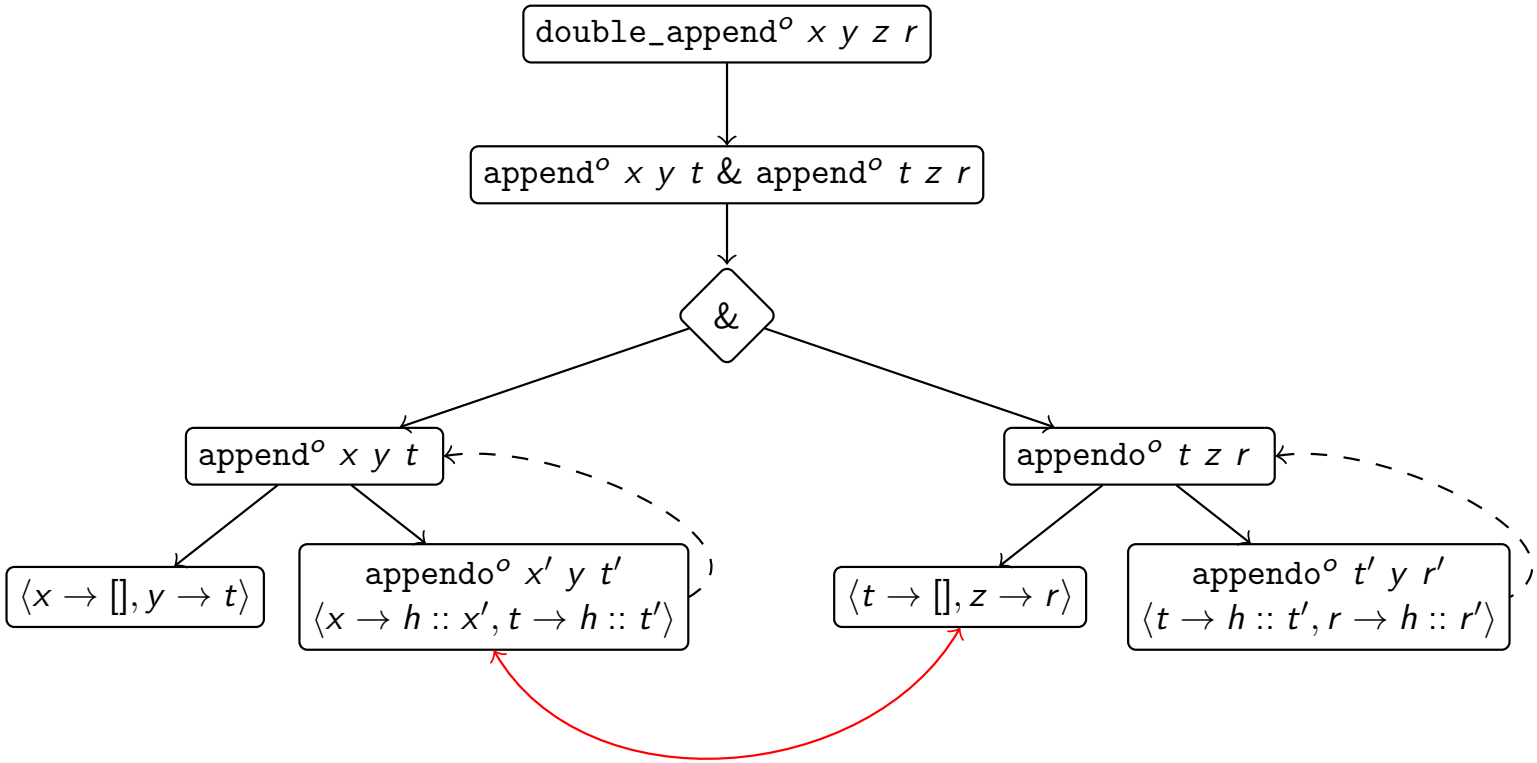
```
let rec appendᵒ x y r =
  ocanren {
    (x ≡ [] & y ≡ r) |
    (fresh h x' r' in
      x ≡ h :: x' &
      appendᵒ x' y r' &
      r ≡ h :: r')}
```

Process tree construction

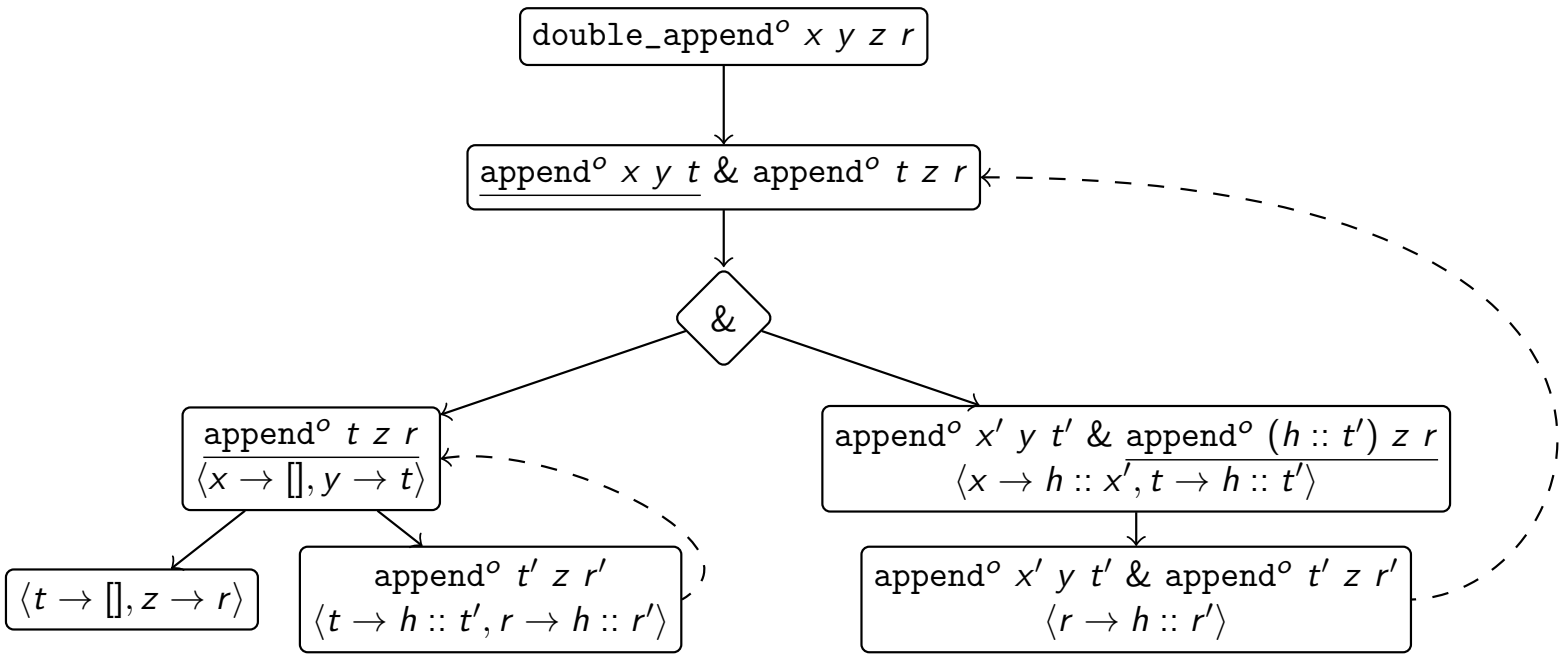


Residualization

Partial Deduction



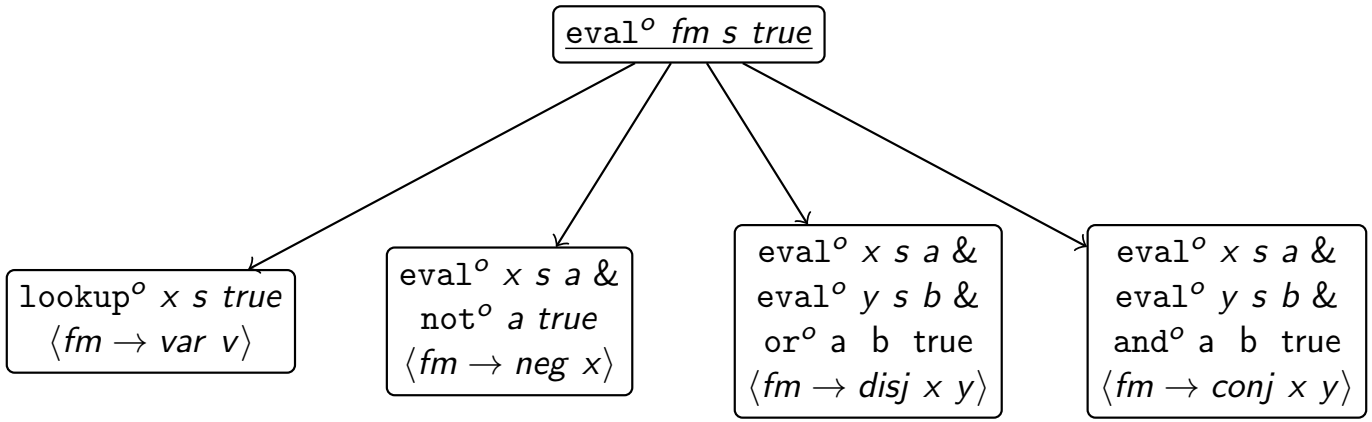
Conjunctive Partial Deduction



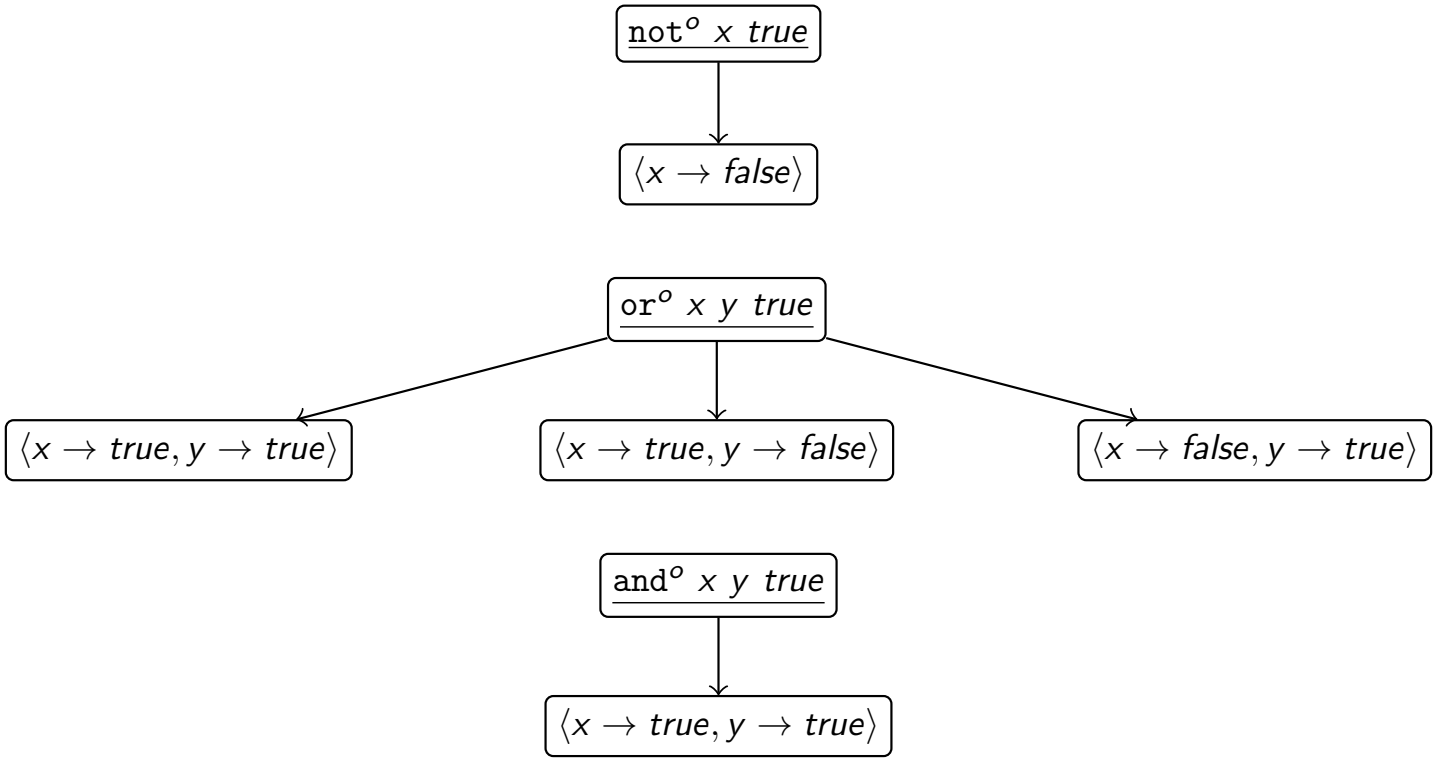
Evaluator of Logic Formulas

```
let rec eval^o fm s r =
  ocanren { fresh v x y a b in
    (fm ≡ var v & lookup^o v s r) |
    (fm ≡ neg x & eval^o x s a & not^o a r) |
    (fm ≡ conj x y & eval^o x s a & eval^o y s b & and^o a b r) |
    (fm ≡ disj x y & eval^o x s a & eval^o y s b & oro^o a b r)
```

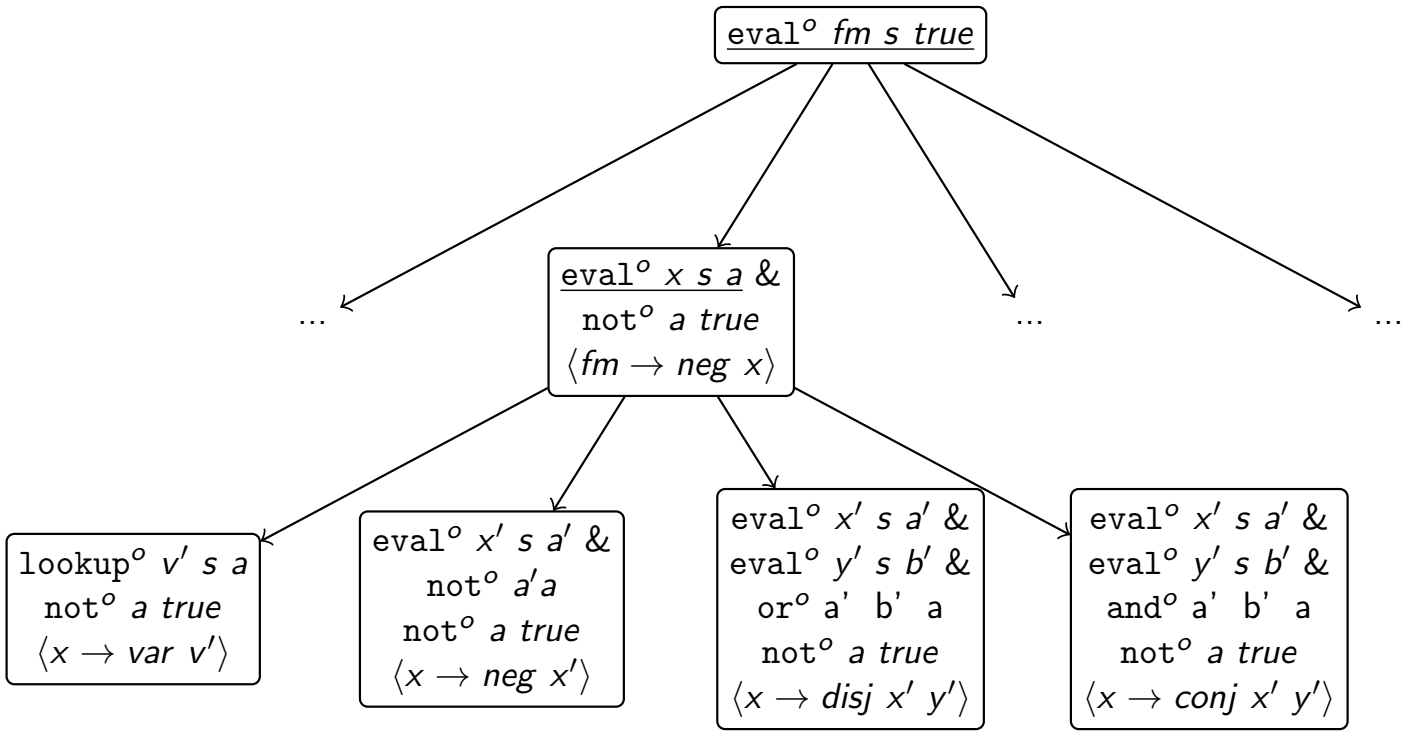
Evaluator of Logic Formulas: Unfolding



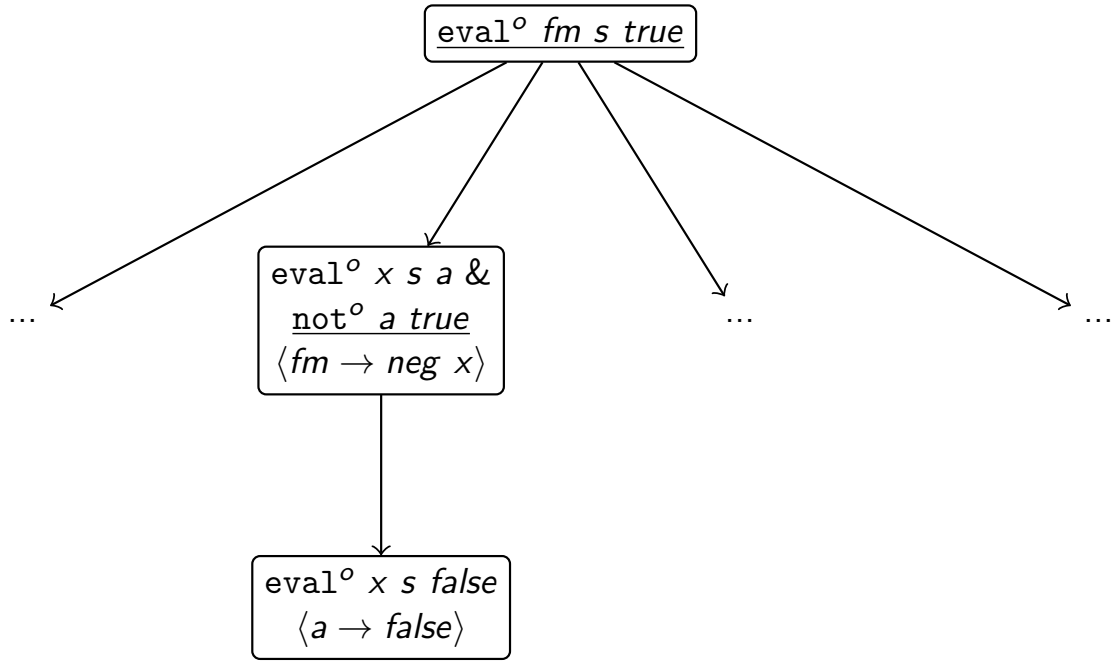
Boolean Connectives



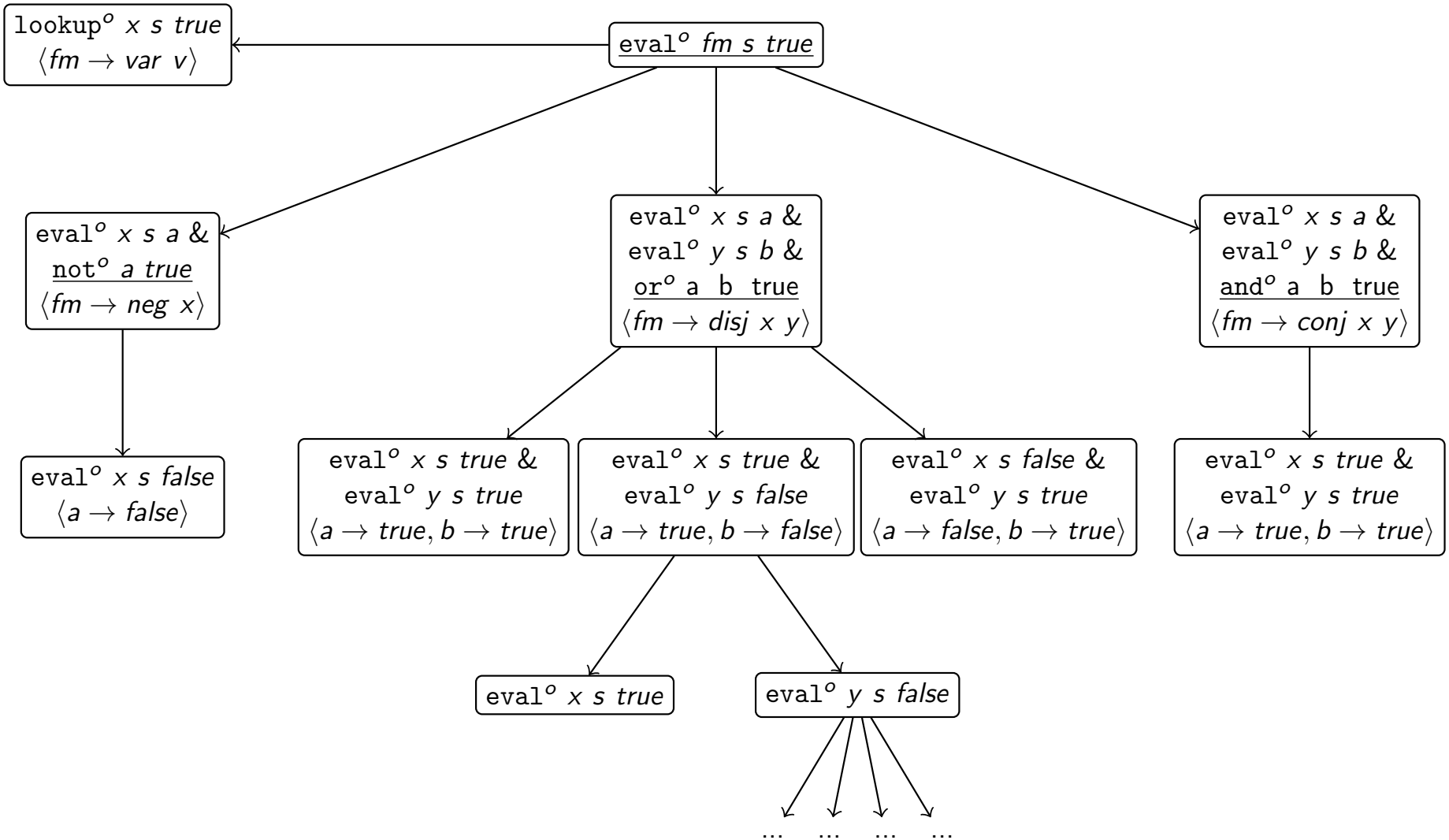
Evaluator of Logic Formulas: Unfolding 2



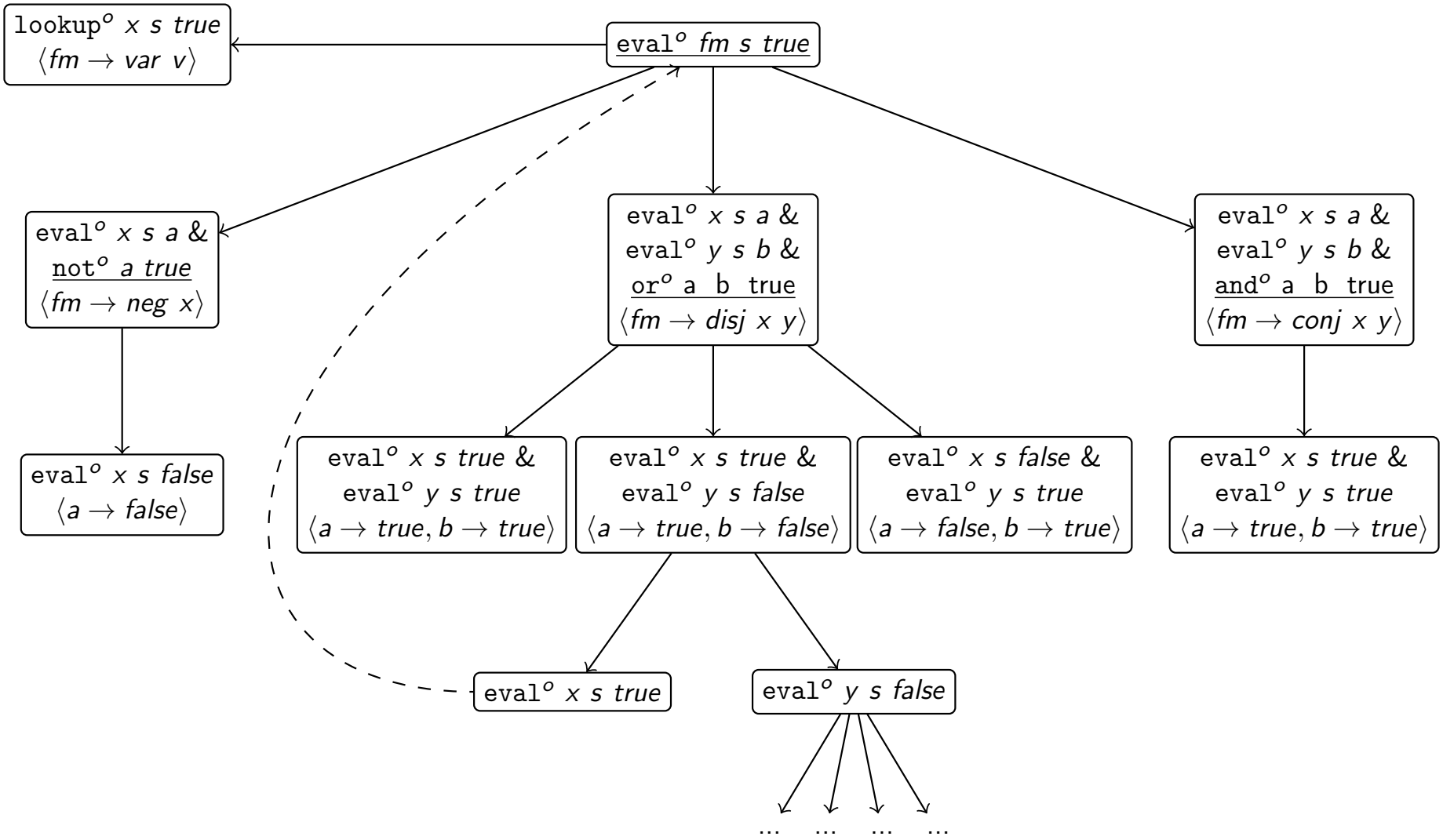
Evaluator of Logic Formulas: Unfolding 3



Evaluator of Logic Formulas: ConsPD

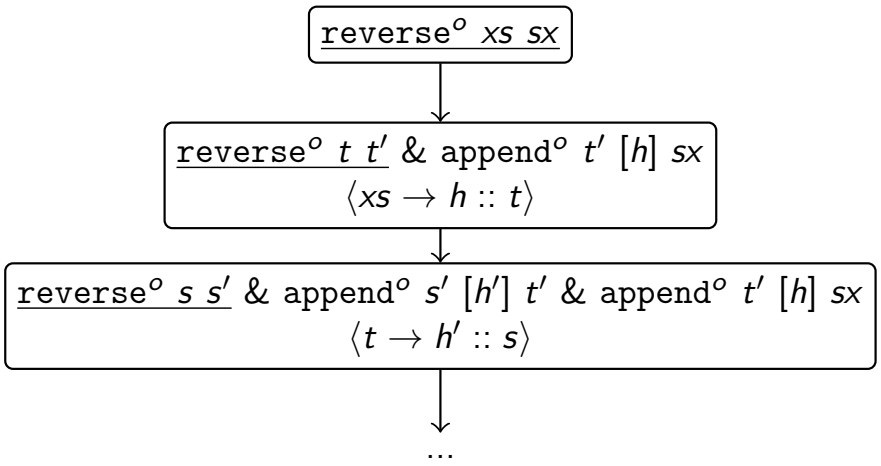


Evaluator of Logic Formulas: ConsPD

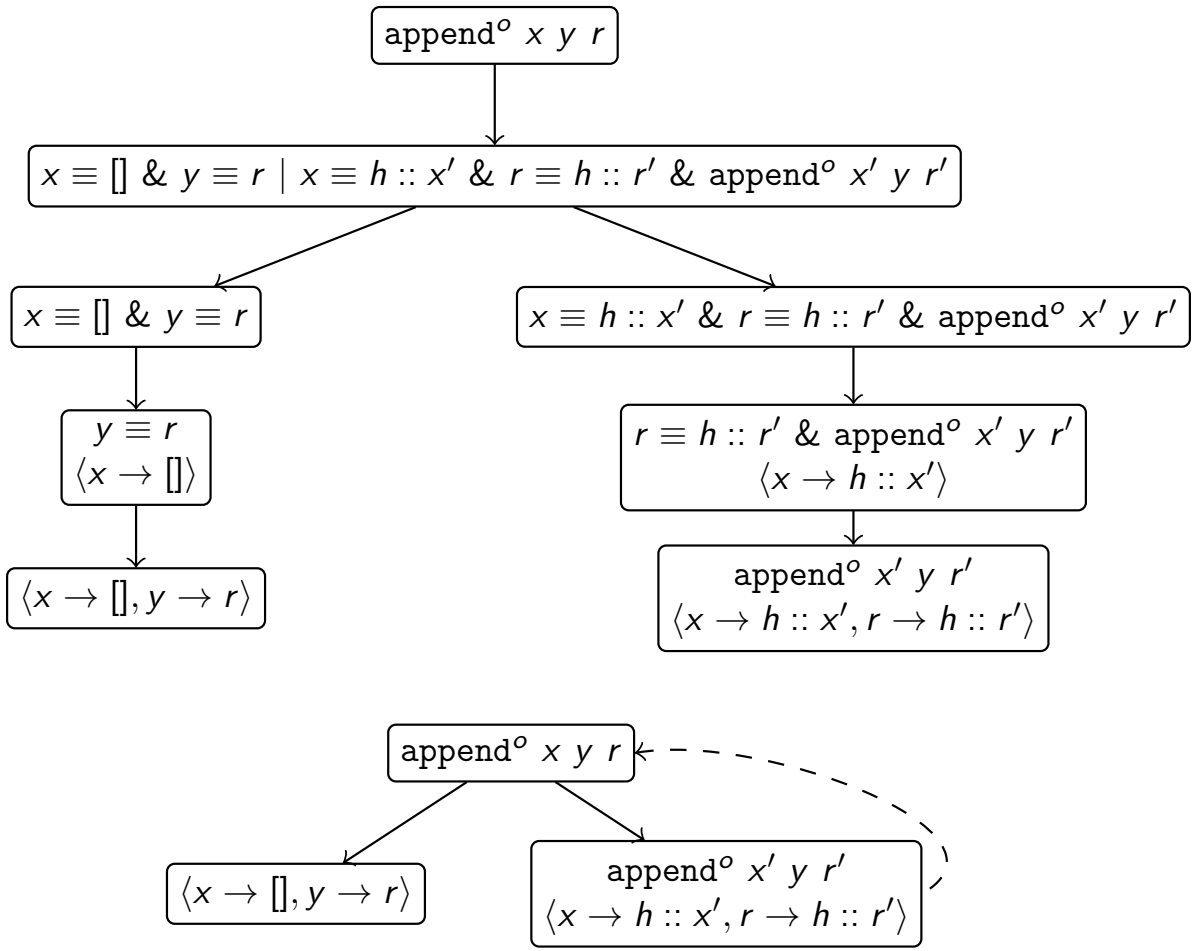


reverse^o

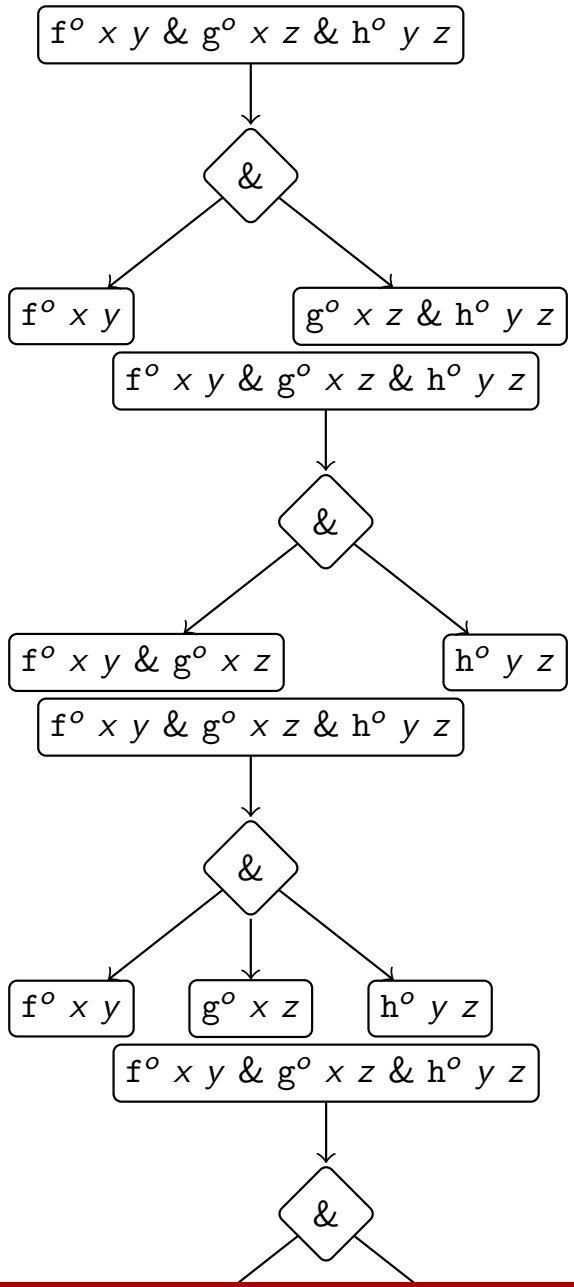
```
let rec reverse^o xs sx =
  ocanren {
    (xs ≡ [] & sx ≡ []) |
    (fresh h t t' in
      xs ≡ h :: t &
      reverse^o t t' &
      append^o t' [h] sx}
```



Unfolding

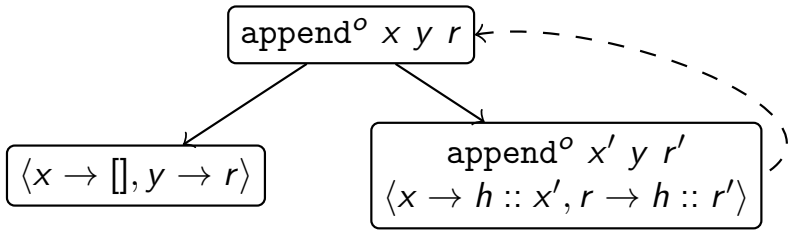

append^o x y r
$x \equiv [] \;\&\; y \equiv r \mid x \equiv h :: x' \;\&\; r \equiv h :: r' \;\&\; \text{append}^o\ x'\ y\ r'$
$x \equiv [] \;\&\; y \equiv r$
$x \equiv h :: x' \;\&\; r \equiv h :: r' \;\&\; \text{append}^o\ x'\ y\ r'$
$y \equiv r$
$\langle x \to [] \rangle$
$r \equiv h :: r' \;\&\; \text{append}^o\ x'\ y\ r'$
$\langle x \to h :: x' \rangle$
$\langle x \to [], y \to r \rangle$
append^o x' y r'
$\langle x \to h :: x', r \to h :: r' \rangle$
append^o x y r
$\langle x \to [], y \to r \rangle$
append^o x' y r'
$\langle x \to h :: x', r \to h :: r' \rangle$

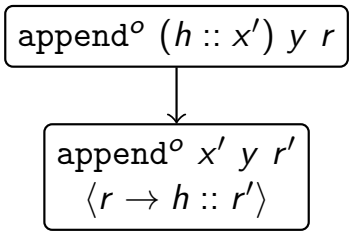
Split


f^o x y & g^o x z & h^o y z
&
f^o x y
g^o x z & h^o y z
f^o x y & g^o x z & h^o y z
&
f^o x y & g^o x z
h^o y z
f^o x y & g^o x z & h^o y z
&
f^o x y
g^o x z
h^o y z
f^o x y & g^o x z & h^o y z
&

Conservative Partial Deduction

Branching Heuristics





Evaluation

Evaluator of Logic Formulas

Evaluator of Logic Formulas: Order of Calls

:

Evaluator of Logic Formulas: Compexity of Relations

Evaluator of Logic Formulas: Results

Unification

Path Search

Evaluation Results

	last	plain	unify	isPath
Original	1.06s	1.84s	—	—
CPD	—	1.13s	14.12s	3.62s
ConsPD	0.93s	0.99s	0.96s	2.51s
Branching	3.11s	7.53s	3.53s	0.54s

Table: Evaluation results

Conclusion

- Conservative Partial Deduction
  - Less-branching heuristics
- Evaluation shows some improvement, but not for every query
- Models to predict performance can help