Verifiying the Top-Down Solver in Isabelle

Sarah Tilscher, Yannick Stade, Helmut Seidl

Technical University of Munich

AVM 2023

Constraint Systems

$\mathbb{D} := 2^{\{a,b,c\}}, \ \sqsubseteq := \subseteq$

$$\begin{aligned} y &\sqsupseteq \{a\} \cup z \\ z &\sqsupseteq y \cup w \\ w &\sqsupseteq \{c\} \end{aligned}$$

Constraint Systems

$\mathbb{D} := 2^{\{a,b,c\}}, \ \sqsubseteq := \subseteq$

$$
\begin{aligned}
y &\sqsupseteq \{a\} \cup z \\
z &\sqsupseteq y \cup w \\
w &\sqsupseteq \{c\}
\end{aligned}
$$

Solve a system of inequalities:

$$x_i \sqsupseteq f_i(x_1, \ldots), \quad i = 1, \ldots$$

The Top-Down Solver

Recursive evaluation:

- starts with an interesting unknown
- descends to query influencing unknowns in right-hand side
- iterates on a queried unknown until local fixpoint is found

The Top-Down Solver

Recursive evaluation:

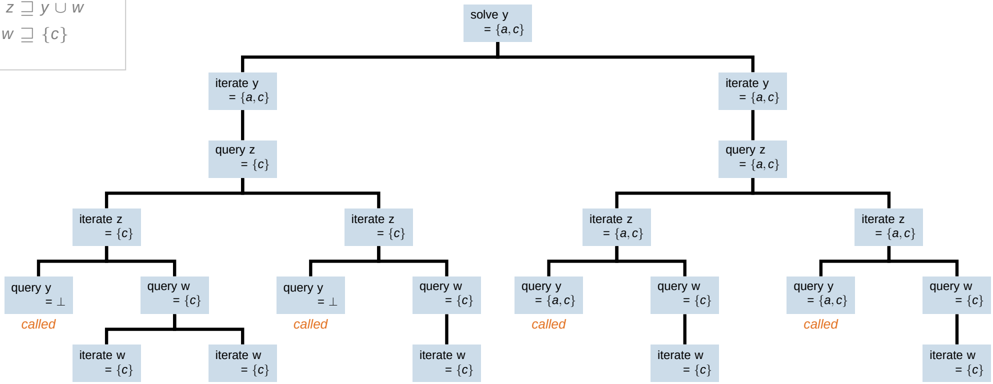
- starts with an interesting unknown
- descends to query influencing unknowns in right-hand side
- iterates on a queried unknown until local fixpoint is found

Self-observation:

- map σ from unknowns to their last evaluated value
- set of called unknowns: to detect recursive dependencies
- set of stable unknowns: value can be looked up without re-iteration
- map of influenced unknowns: re-evaluate when value changes

The Trace of the Solver

$$y \sqsupseteq \{a\} \cup z$$
$$z \sqsupseteq y \cup w$$
$$w \sqsupseteq \{c\}$$



Proving Partial Correctness

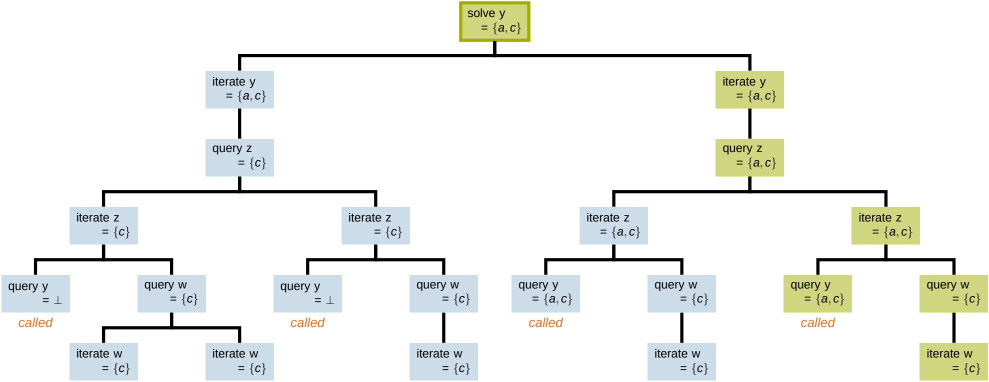
call to solve terminates and $\textit{solve}\ x = (\textit{xd}, \sigma)$

$$\Downarrow$$

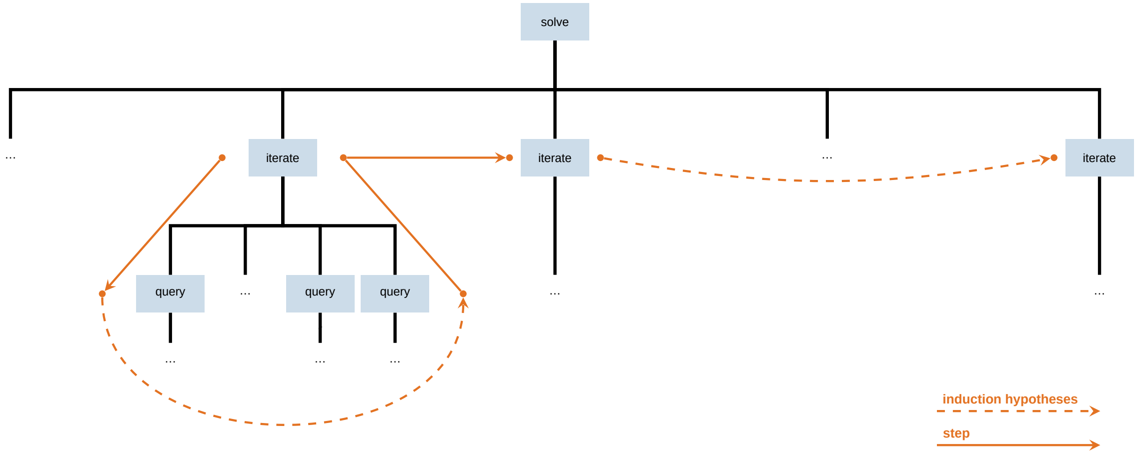
$$\forall y \in \textit{reach}\ x. \quad \textit{eq}\ y\ \sigma = \sigma\ y$$

reach: set of unknowns evaluated during the final iteration

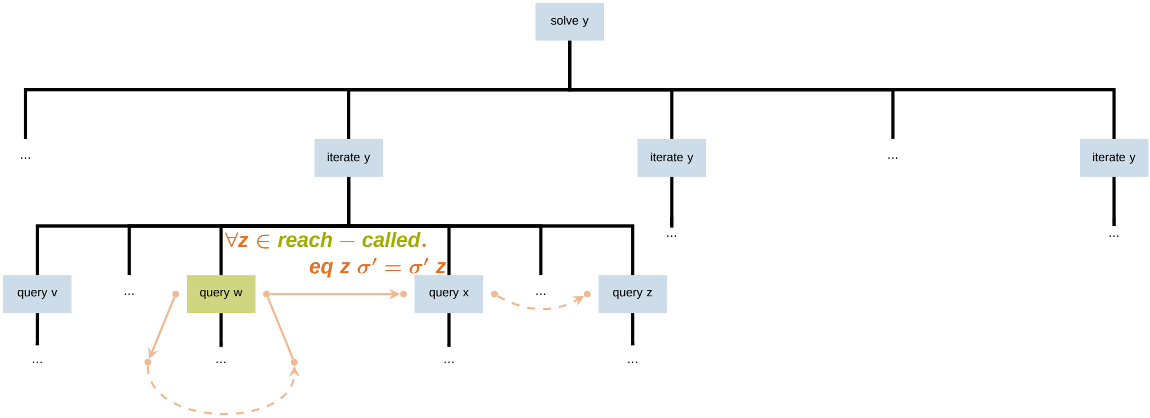
Reach


solve y
= {a, c}
iterate y
= {a, c}
iterate y
= {a, c}
query z
= {c}
query z
= {a, c}
iterate z
= {c}
iterate z
= {c}
iterate z
= {a, c}
iterate z
= {a, c}
query y
= ⊥
called
query w
= {c}
query y
= ⊥
called
query w
= {c}
query y
= {a, c}
called
query w
= {c}
query y
= {a, c}
called
query w
= {c}
iterate w
= {c}
iterate w
= {c}
iterate w
= {c}
iterate w
= {c}
iterate w
= {c}

Induction over the Trace

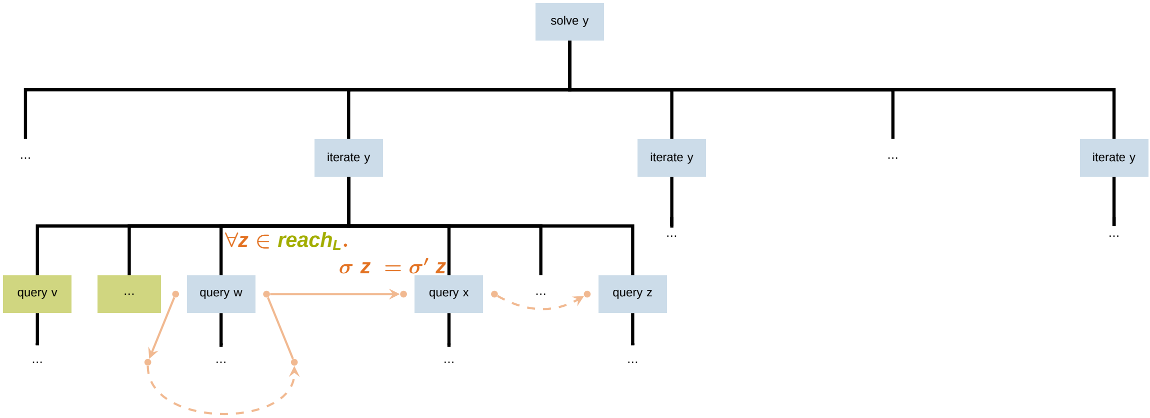


Invariants



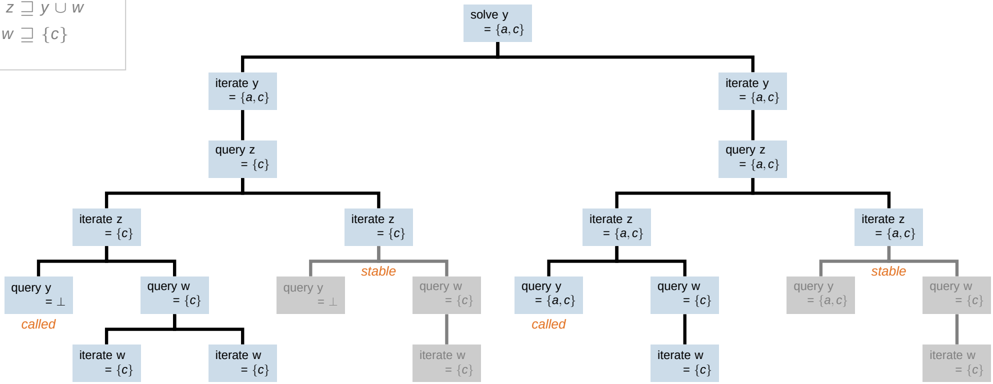
solve y
...
iterate y
iterate y
...
iterate y
...
...
$\forall z \in reach - called.$
$eq\ z\ \sigma' = \sigma'\ z$
query v
...
query w
query x
...
query z
...
...
...
...

Invariants


solve y
...
iterate y
iterate y
...
iterate y
...
...
∀z ∈ reach_L .
σ z = σ′ z
query v
...
query w
query x
...
query z
...
...
...
...

Ongoing Work: Record Stable Unknowns

$$y \sqsupseteq \{a\} \cup z$$
$$z \sqsupseteq y \cup w$$
$$w \sqsupseteq \{c\}$$



Summary

- formalized the top-down solver in Isabelle
- proved partial correctness for the simplified Top-Down Solver
  - by induction over trace
  - with invariants about its state of computation
- ongoing: show equivalence to vanilla Top-Down Solver

The Top-Down Solver - simplified

```
solve x = iterate x {x} empty_map
```

The Top-Down Solver - simplified

```
solve x = iterate x {x} empty_map
query (Answer d) called σ = (d, σ)
query (Query x f) called σ = (
```

```
iterate x called σ = (
```

The Top-Down Solver - simplified

```
solve x = iterate x {x} empty_map
query (Answer d) called σ = (d, σ)
query (Query x f) called σ = (
    let (xd, σ) =
      if x ∉ called then
        iterate x (insert x called) σ



    in


iterate x called σ = (
```

The Top-Down Solver - simplified

```
solve x = iterate x {x} empty_map
query (Answer d) called σ = (d, σ)
query (Query x f) called σ = (
    let (xd, σ) =
      if x ∉ called then
        iterate x (insert x called) σ
      else
        (fmlup σ x, σ)
    in


iterate x called σ = (
```

The Top-Down Solver - simplified

```
solve x = iterate x {x} empty_map
query (Answer d) called σ = (d, σ)
query (Query x f) called σ = (
    let (xd, σ) =
      if x ∉ called then
        iterate x (insert x called) σ
      else
        (fmlup σ x, σ)
    in
    query (f xd) called σ
iterate x called σ = (
```

The Top-Down Solver - simplified

```
solve x = iterate x {x} empty_map
query (Answer d) called σ = (d, σ)
query (Query x f) called σ = (
    let (xd, σ) =
      if x ∉ called then
        iterate x (insert x called) σ
      else
        (fmlup σ x, σ)
    in
    query (f xd) called σ
iterate x called σ = (
    let (d_new, σ) = query (T x) called σ in
    if d_new = fmlup σ x then
      (d_new, σ)
```

The Top-Down Solver - simplified

```
solve x = iterate x {x} empty_map

query (Answer d) called σ = (d, σ)

query (Query x f) called σ = (
    let (xd, σ) =
      if x ∉ called then
        iterate x (insert x called) σ
      else
        (fmlup σ x, σ)
    in
    query (f xd) called σ

iterate x called σ = (
    let (d_new, σ) = query (T x) called σ in
    if d_new = fmlup σ x then
      (d_new, σ)
    else
      iterate x called (fmupd x d_new σ))
```