Possibly Infinite Trees of Finite Degree

Working towards König's Lemma

Lukas Gerlach

Knowledge-Based Systems Group, TU Dresden, Germany

13.03.2025

Background / Short Motivation

My work revolves around the chase algorithm on existential rules.

(Database Theory and Knowledge Representation)

Background / Short Motivation

My work revolves around the chase algorithm on existential rules.

(Database Theory and Knowledge Representation)

$$R(x, y) \to \exists z. R(y, z) \vee R(y, x)$$

Background / Short Motivation

My work revolves around the chase algorithm on existential rules.

(Database Theory and Knowledge Representation)

$$R(x, y) \to \exists z.R(y, z) \vee R(y, x)$$

With the fact $R(a, b)$, the chase yields an infinite tree of fact sets.

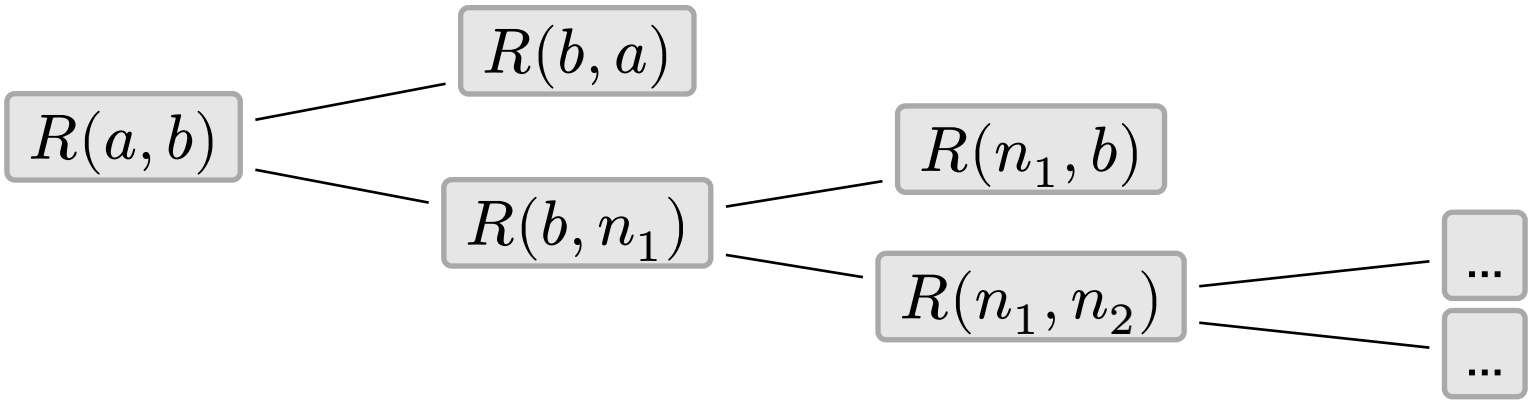
Background / Short Motivation

My work revolves around the chase algorithm on existential rules.

(Database Theory and Knowledge Representation)

$$R(x, y) \to \exists z.R(y, z) \vee R(y, x)$$

With the fact $R(a, b)$, the chase yields an infinite tree of fact sets.



What about inductive trees?

What about inductive trees?

```
inductive BinaryTree a with
```

What about inductive trees?

```
inductive BinaryTree a with
| leaf a -> BinaryTree a
```

What about inductive trees?

```
inductive BinaryTree a with
| leaf a -> BinaryTree a
| inner a -> BinaryTree a -> BinaryTree a -> BinaryTree a
```

What about inductive trees?

```
inductive BinaryTree a with
| leaf a -> BinaryTree a
| inner a -> BinaryTree a -> BinaryTree a -> BinaryTree a
```

Inductive types express the least fixed point, i.e. the minimal set of trees obtainable from the two constructors.

What about inductive trees?

```
inductive BinaryTree a with
| leaf a -> BinaryTree a
| inner a -> BinaryTree a -> BinaryTree a -> BinaryTree a
```

Inductive types express the least fixed point, i.e. the minimal set of trees obtainable from the two constructors. We get all leaves, then for each pair of existing trees, we construct an inner tree.

What about inductive trees?

```
inductive BinaryTree a with
| leaf a -> BinaryTree a
| inner a -> BinaryTree a -> BinaryTree a -> BinaryTree a
```

Inductive types express the least fixed point, i.e. the minimal set of trees obtainable from the two constructors. We get all leaves, then for each pair of existing trees, we construct an inner tree. This gives us trees of ever increasing size but they are all finite.

What about inductive trees?

```
inductive BinaryTree a with
| leaf a -> BinaryTree a
| inner a -> BinaryTree a -> BinaryTree a -> BinaryTree a
```

Inductive types express the least fixed point, i.e. the minimal set of trees obtainable from the two constructors. We get all leaves, then for each pair of existing trees, we construct an inner tree. This gives us trees of ever increasing size but they are all finite.

We want the greatest fixed point instead, i.e. a coinductive definition!

It's time for some code!

1.
2.
3.
4.
5.

https://dmfa.dev/leaning-in-2025

It's time for some code!

1. Take a hint from infinite lists.
2.
3.
4.
5.

https://dmfa.dev/leaning-in-2025

It's time for some code!

1. Take a hint from infinite lists.
2. Define possibly infinite lists.
3.
4.
5.

https://dmfa.dev/leaning-in-2025

It's time for some code!

1. Take a hint from infinite lists.
2. Define possibly infinite lists.
3. Define infinite trees.
4.
5.

https://dmfa.dev/leaning-in-2025

It's time for some code!

1. Take a hint from infinite lists.
2. Define possibly infinite lists.
3. Define infinite trees.
4. Define trees with finite degree.
5.

https://dmfa.dev/leaning-in-2025

It's time for some code!

1. Take a hint from infinite lists.
2. Define possibly infinite lists.
3. Define infinite trees.
4. Define trees with finite degree.
5. Prove König's Lemma.

https://dmfa.dev/leaning-in-2025

It's time for some code!

1. Take a hint from infinite lists.
2. Define possibly infinite lists.
3. Define infinite trees.
4. Define trees with finite degree.
5. Prove König's Lemma.

König's Lemma: (Special Case)

It's time for some code!

1. Take a hint from infinite lists.
2. Define possibly infinite lists.
3. Define infinite trees.
4. Define trees with finite degree.
5. Prove König's Lemma.

König's Lemma: (Special Case)
If each branch of a tree with finite degree is finite,

It's time for some code!

1. Take a hint from infinite lists.
2. Define possibly infinite lists.
3. Define infinite trees.
4. Define trees with finite degree.
5. Prove König's Lemma.

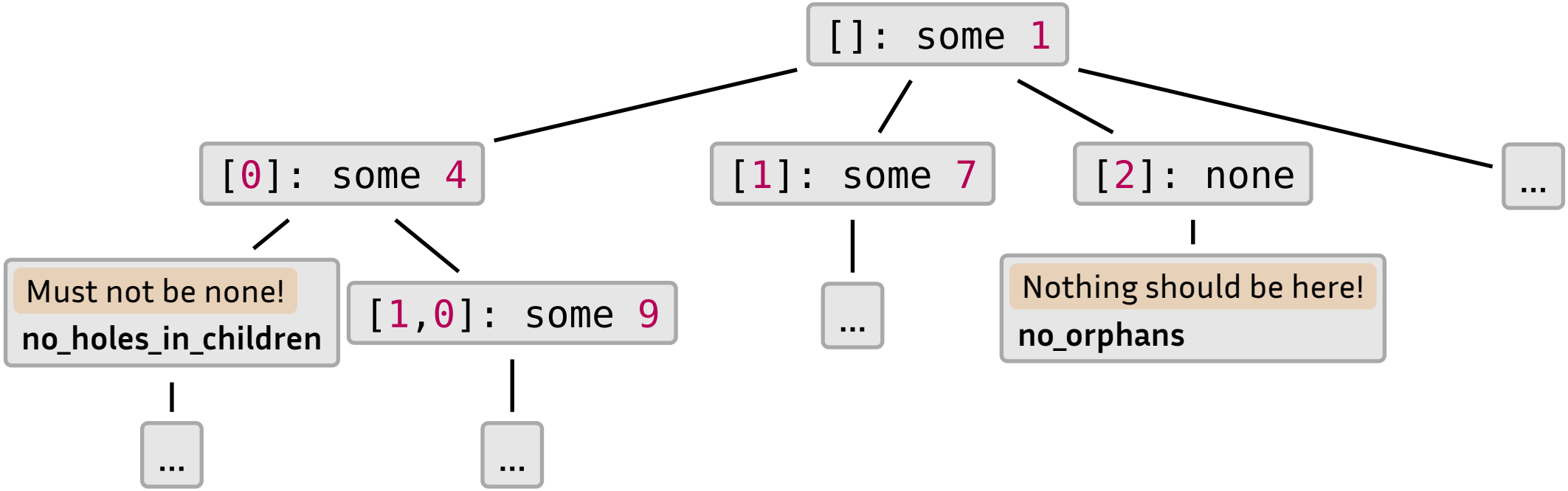
König's Lemma: (Special Case)
If each branch of a tree with finite degree is finite,
then there are only finitely many branches.

Possibly Infinite Tree - The Idea

```
def PossiblyInfiniteTree (a : Type u) := List Nat -> Option a
```

Possibly Infinite Tree - The Idea

```
def PossiblyInfiniteTree (a : Type u) := List Nat -> Option a
```



That's all!

Thank you so much for having me! 🤓

I hope you can get all your

`goals accomplished` 🎉

Feel free to reach out in Lean Zulipchat or via mail:

`lukas.gerlach@tu-dresden.de`

`hi@monsterkrampe.dev`