Chapter 3

Fixpoints for representation predicates

In this chapter we show how non-structurally recursive representation predicates can be defined using least fixpoints. In section 3.1 we explain why it is hard to define non-structurally recursive predicates and generally explain the approach that is taken. Next, in section 3.2 we show the way least fixpoints are defined in Iris. Lastly, in section 3.3 we explain the improvements we made to the approach of Iris in order for the process to be automated.

3.1 Problem statement

The logic and definitions we are describing are embedded in the proof assistant Coq. This imposes a restriction on the logic. We are not allowed to have non-structurally recursive separation logic predicates.

The candidate argument for structural recursion in isMLLwould be the list of values used to represent the MLL. However, this does not work given the second case of the structural recursion.

$$\mathsf{isMLL}\, hd\, \vec{v} = \cdots \vee (\exists \ell, v', tl.\, hd = \texttt{some}\, l * l \mapsto (v', \texttt{true}, tl) * \mathsf{isMLL}\, tl\, \vec{v}) \vee \cdots$$

Here the list of values is passed straight onto the recursive call to isMLL. Thus, it is not structurally recursive.

We need another approach to define non-structurally recursive predicates such as these. Iris has several approaches to fix this problem. The most widely applicable one takes an approach inspired by the Knaster-Tarski fixpoint theorem[Tar55]. Given a monotone functor on predicates, there exists a least fixpoint of this functor. We can now choose a functor such that the fixpoint corresponds to the recursive predicate we wanted to design. This procedure is explained thoroughly in the next section, section 3.2.

3.2 Least fixpoint in Iris

To define a least fixpoint in Iris the first step is to have a monotone functor.

Definition 3.1: Monotone functor

Predicate $\mathsf{F} \colon (A \to \mathit{iProp}) \to A \to \mathit{iProp}$ is monotone when for any $\Phi, \Psi \colon A \to \mathit{iProp}$, it holds that

$$\Box(\forall y.\, \Phi\, y \mathrel{-\!\!*} \Psi\, y) \vdash \forall x.\, \mathsf{F}\, \Phi\, x \mathrel{-\!\!*} \mathsf{F}\, \Psi\, x$$

In other words, it is monotone in its first argument.

This definition of monotone follows the definition of monotone in other fields with one exception. The assumption has an additional restriction, it has to be persistent. The persistence is necessary since F could use its monotone argument multiple times.

Example 3.2

Take the following functor.

$$\begin{aligned}\mathsf{F}\, \Phi\, v \triangleq{}& (v = \mathbf{none}) \lor \\ & (\exists \ell_1, \ell_2, v_1, v_2.\, v = \mathbf{some}(\ell_1, \ell_2) * \ell_1 \mapsto v_1 * \ell_2 \mapsto v_2 * \Phi\, v_1 * \Phi\, v_2)\end{aligned}$$

This is the functor for binary trees. The value v is either empty, and we have an empty tree. Or v contains two locations, for the two branches of the tree. Each location points to a value and Φ is holds for both of these values. The fixpoint, as is discussed in theorem 3.3, of this functor holds for a value containing a binary tree. However, before we can take the fixpoint we have to prove it is monotone.

$$\Box(\forall w.\, \Phi\, w \mathrel{-\!\!*} \Psi\, w) \vdash \forall v.\, \mathsf{F}\, \Phi\, v \mathrel{-\!\!*} \mathsf{F}\, \Psi\, v$$

Proof. We start by introducing v and the wand.

$$\Box(\forall w.\, \Phi\, w \mathrel{-\!\!*} \Psi\, w) * \mathsf{F}\, \Phi\, v \vdash \mathsf{F}\, \Psi\, v$$

We now unfold the definition of F and eliminate and introduce the disjunction, resulting in two statements to prove.

$$\Box(\forall w.\, \Phi\, w \mathrel{-\!\!*} \Psi\, w) * v = \mathbf{none} \vdash v = \mathbf{none}$$

$$\Box(\forall w.\, \Phi\, w \mathrel{-\!\!*} \Psi\, w) * \left(\exists \ell_1, \ell_2, v_1, v_2.\ \begin{array}{l} v = \mathbf{some}(\ell_1, \ell_2) * \ell_1 \mapsto v_1 * \\ \ell_2 \mapsto v_2 * \Phi\, v_1 * \Phi\, v_2 \end{array}\right) \vdash$$
$$\left(\exists \ell_1, \ell_2, v_1, v_2.\ \begin{array}{l} v = \mathbf{some}(\ell_1, \ell_2) * \ell_1 \mapsto v_1 * \\ \ell_2 \mapsto v_2 * \Psi\, v_1 * \Psi\, v_2 \end{array}\right)$$

The first statement holds directly. For the second statement we eliminate the existentials in the assumption and use the created variables to introduce the existentials in the conclusion.

$$\Box(\forall w.\, \Phi\, w \mathrel{-\!\!*} \Psi\, w) * \begin{array}{l} v = \texttt{some}(\ell_1, \ell_2) * \\ \ell_1 \mapsto v_1 * \ell_2 \mapsto v_2 * \\ \Phi\, v_1 * \Phi\, v_2 \end{array} \vdash \begin{array}{l} v = \texttt{some}(\ell_1, \ell_2) * \\ \ell_1 \mapsto v_1 * \ell_2 \mapsto v_2 * \\ \Psi\, v_1 * \Psi\, v_2 \end{array}$$

Any sub propositions that occur both on the left and right-hand side are canceled out using $*$-MONO.

$$\Box(\forall w.\, \Phi\, w \mathrel{-\!\!*} \Psi\, w) * \Phi\, v_1 * \Phi\, v_2 \vdash \Psi\, v_1 * \Psi\, v_2$$

We want to split the conclusion and premise in two, such that we get the following statements, with $i \in \{1, 2\}$.

$$\Box(\forall w.\, \Phi\, w \mathrel{-\!\!*} \Psi\, w) * \Phi\, v_i \vdash \Psi\, v_i$$

To achieve this split, we duplicate the persistent premise and then split using $*$-MONO again. Both these statements hold trivially. □

In the previous proof it was essential that the premise of monotonicity is persistent. This occurs any time we have a data structure with more than one branch.

Now that we have a definition of a functor, we can prove that a least fixpoint of a monotone functor always exists.

Theorem 3.3: Least fixpoint

Given a monotone functor $\mathsf{F}\colon (A \to iProp) \to A \to iProp$, there exists a least fixpoint $\mu\mathsf{F}\colon A \to iProp$ such that

1. The bidirectional unfolding property holds

$$\mu\mathsf{F}\, x \dashv\vdash \mathsf{F}\,(\mu\mathsf{F})\, x$$

2. The iteration property holds

$$\Box\, \forall y.\, \mathsf{F}\,\Phi\, y \mathrel{-\!\!*} \Phi\, y \vdash \forall x.\, \mu\mathsf{F}\, x \mathrel{-\!\!*} \Phi\, x$$

Question: Maybe move this proof to the appendix, it is not very interesting?

Proof. Given a monotone functor $\mathsf{F}\colon (A \to iProp) \to A \to iProp$ we define $\mu\mathsf{F}$ as

$$\mu\mathsf{F}\, x \triangleq \forall \Phi.\ \Box(\forall y.\, \mathsf{F}\,\Phi\, y \mathrel{-\!\!*} \Phi\, y) \mathrel{-\!\!*} \Phi\, x$$

We now prove the two properties of the least fixpoint

1. We start with proving this right to left, then using the result, prove left to right.

   R-L We first unfold the definition of $\mu\mathsf{F}\,x$.

   $$\mathsf{F}\,(\mu\mathsf{F})\,x \vdash \forall\Phi.\ \Box(\forall y.\, \mathsf{F}\,\Phi\,y \twoheadrightarrow \Phi\,y) \twoheadrightarrow \Phi\,x$$

   Next we introduce Φ and the wand.

   $$\mathsf{F}\,(\mu\mathsf{F})\,x * \Box(\forall y.\, \mathsf{F}\,\Phi\,y \twoheadrightarrow \Phi\,y) \vdash \Phi\,x$$

   We now apply $\Box(\forall y.\, \mathsf{F}\,\Phi\,y \twoheadrightarrow \Phi\,y)$ to $\Phi\,x$.

   $$\mathsf{F}\,(\mu\mathsf{F})\,x * \Box(\forall y.\, \mathsf{F}\,\Phi\,y \twoheadrightarrow \Phi\,y) \vdash \mathsf{F}\,\Phi\,x$$

   We revert $\mathsf{F}\,(\mu\mathsf{F})\,x$ and apply the monotonicity of F.

   $$\Box(\forall y.\, \mathsf{F}\,\Phi\,y \twoheadrightarrow \Phi\,y) \vdash \mu\mathsf{F}\,x \twoheadrightarrow \Phi\,x$$

   After introducing the wand and applying the definition of $\mu\mathsf{F}$ we get

   $$(\forall\Phi.\ \Box(\forall y.\, \mathsf{F}\,\Phi\,y \twoheadrightarrow \Phi\,y) \twoheadrightarrow \Phi\,x) * \Box(\forall y.\, \mathsf{F}\,\Phi\,y \twoheadrightarrow \Phi\,y) \vdash \Phi\,x$$

   This statement holds by application of the first assumption.

   L-R We again first apply the definition of $\mu\mathsf{F}$.

   $$\forall\Phi.\ \Box(\forall y.\, \mathsf{F}\,\Phi\,y \twoheadrightarrow \Phi\,y) \twoheadrightarrow \Phi\,x \vdash \mathsf{F}\,(\mu\mathsf{F})\,x$$

   We apply the assumption with $\Phi = \mathsf{F}\,(\mu\mathsf{F})$ resulting in the following statement after introductions

   $$\mathsf{F}\,(\mathsf{F}\,(\mu\mathsf{F}))\,x \vdash \mathsf{F}\,(\mu\mathsf{F})\,x$$

   This holds because of monotonicity of F and the above proved property.

2. This follows directly from unfolding the definition of $\mu\mathsf{F}$. $\Box$

The first property of theorem 3.3, unfolding, defines that the least fixpoint is a fixpoint. The second property of theorem 3.3, iteration, ensures that this fixpoint is the least of the possible fixpoints. The iteration property is a weaker version of the induction principle. The induction hypothesis during iteration is weaker. It only ensures that Φ holds under F. Full induction requires that we also know that the fixpoint holds under F in the induction hypothesis.

Lemma 3.4

Given a monotone predicate $\mathsf{F}\colon (A \to \mathit{iProp}) \to (A \to \mathit{iProp})$, it holds that

$$\Box(\forall x.\, \mathsf{F}\,(\lambda y.\, \Phi\, y \wedge \mu\mathsf{F}\, y)\, x \mathrel{-\!\!*} \Phi\, x) \mathrel{-\!\!*} \forall x.\, \mu\mathsf{F}\, x \mathrel{-\!\!*} \Phi\, x$$

This lemma follows from monotonicity and the least fixpoint properties. We can now use the above steps to define isMLL

Example 3.5: Iris least fixpoint of isMLL

We want to create a least fixpoint such that it has the following inductive property.

$$\begin{aligned}\mathsf{isMLL}\, hd\, \vec{v} = \;& hd = \mathbf{none} * \vec{v} = [\,] \vee \\ & (\exists \ell, v', tl.\, hd = \mathbf{some}\, l * l \mapsto (v', \mathbf{true}, tl) * \mathsf{isMLL}\, tl\, \vec{v}) \vee \\ & \left(\exists \ell, v', \vec{v}'', tl.\, \begin{array}{l} hd = \mathbf{some}\, l * l \mapsto (v', \mathbf{false}, tl) * \\ \vec{v} = v' :: \vec{v}'' * \mathsf{isMLL}\, tl\, \vec{v}'' \end{array}\right)\end{aligned}$$

The first step is creating the functor. We do this by adding an argument to isMLLtransforming it into a functor. We then substitute any recursive calls to $\mathsf{isMLL}$with this argument.

$$\begin{aligned}\mathsf{isMLL}_{\mathsf{F}}\, \Phi\, hd\, \vec{v} \triangleq \;& hd = \mathbf{none} * \vec{v} = [\,] \vee \\ & (\exists \ell, v', tl.\, hd = \mathbf{some}\, l * l \mapsto (v', \mathbf{true}, tl) * \Phi\, tl\, \vec{v}) \vee \\ & \left(\exists \ell, v', \vec{v}'', tl.\, \begin{array}{l} hd = \mathbf{some}\, l * l \mapsto (v', \mathbf{false}, tl) * \\ \vec{v} = v' :: \vec{v}'' * \Phi\, tl\, \vec{v}'' \end{array}\right)\end{aligned}$$

This has created a functor, $\mathsf{isMLL}_{\mathsf{F}}$. The functor applies the predicate, Φ, on the tail of any possible MLL, while ensuring the head is part of an MLL. Next, we want to prove that $\mathsf{isMLL}_{\mathsf{F}}$ is monotone. However, $\mathsf{isMLL}_{\mathsf{F}}$ has the following type.

$$\mathsf{isMLL}_{\mathsf{F}}\colon (\mathit{Val} \to \mathit{List\, Val} \to \mathit{iProp}) \to \mathit{Val} \to \mathit{List\, Val} \to \mathit{iProp}$$

But, definition 3.1 only works for functors of type

$$\mathsf{F}\colon (A \to \mathit{iProp}) \to A \to \mathit{iProp}$$

This is solved by uncurrying $\mathsf{isMLL}_{\mathsf{F}}$

$$\mathsf{isMLL}'_{\mathsf{F}}\, \Phi\, (hd, \vec{v}) \triangleq \mathsf{isMLL}_{\mathsf{F}}\, \Phi\, hd\, \vec{v}$$

The functor $\mathsf{isMLL}'_{\mathsf{F}}$ now has the type

$$\mathsf{isMLL}'_{\mathsf{F}}\colon (\mathit{Val} \times \mathit{List\, Val} \to \mathit{iProp}) \to \mathit{Val} \times \mathit{List\, Val} \to \mathit{iProp}$$

And we can prove it monotone.

$$\begin{gathered}\Box(\forall (hd, \vec{v}).\, \Phi\, (hd, \vec{v}) \mathrel{-\!\!*} \Psi\, (hd, \vec{v})) \\ \vdash \forall (hd, \vec{v}).\, \mathsf{isMLL}'_{\mathsf{F}}\, \Phi\, (hd, \vec{v}) \mathrel{-\!\!*} \mathsf{isMLL}'_{\mathsf{F}}\, \Psi\, (hd, \vec{v})\end{gathered}$$

Proof. We use a similar proof as in example 3.2. It involves more steps as we have more branches, but the same ideas apply. $\square$

Given that $\mathsf{isMLL}'_{\mathsf{F}}$ is monotone, we now know from theorem 3.3 that the least fixpoint exists of $\mathsf{isMLL}'_{\mathsf{F}}$.

$$\mathsf{isMLL}'\,(hd, \vec{v}) \triangleq \mu(\mathsf{isMLL}'_{\mathsf{F}})\,(hd, \vec{v})$$

To finish the definition of isMLLwe uncurry the created fixpoint

$$\mathsf{isMLL}\,hd\,\vec{v} \triangleq \mathsf{isMLL}'\,(hd, \vec{v})$$

This definition of isMLL has the inductive property as described in section 2.5. That property is the left to right unfolding property. After expanding any currying lemma 3.4 we get the below induction principle for isMLL.

$$\begin{aligned} &\Box(\forall hd, \vec{v}.\, \mathsf{isMLL}_{\mathsf{F}}\,(\lambda hd', \vec{v}'.\, \Phi\, hd'\, \vec{v}' \wedge \mathsf{isMLL}\, hd'\, \vec{v}')\, hd\, \vec{v} \mathrel{-\!\!*} \Phi\, hd\, \vec{v}) \\ \mathrel{-\!\!*}\;\; &\forall hd, \vec{v}.\, \mathsf{isMLL}\, hd\, \vec{v} \mathrel{-\!\!*} \Phi\, hd\, \vec{v} \end{aligned}$$

The induction principle from section 2.5 is also derivable from lemma 3.4. The three cases of the induction principle follow from the disjunctions in $\mathsf{isMLL}_{\mathsf{F}}$.

3.3 Syntactic monotone proof search

As we discussed in chapter 1, the goal of this thesis is to show how to automate the definition of representation predicates from inductive definitions. The major hurdle in this process can be seen in example 3.5, proving a functor monotone. In this section we show how a monotonicity proof can be found by using syntactic proof search.

We take the following strategy. We prove the monotonicity of all the connectives once. We now prove the monotonicity of the functor by making use of the monotonicity of the connectives with which it is built.

Monotone connectives We don't want to uncurry every connective when using that it is monotone, thus we take a different approach on what is monotone. For every connective we give a signature telling us how it is monotone. We show a few of these signatures below.

Connective	Type	Signature
$*$	$iProp \to iProp \to iProp$	$(\mathrel{-\!\!*}) \Longrightarrow (\mathrel{-\!\!*}) \Longrightarrow (\mathrel{-\!\!*})$
$\vee$	$iProp \to iProp \to iProp$	$(\mathrel{-\!\!*}) \Longrightarrow (\mathrel{-\!\!*}) \Longrightarrow (\mathrel{-\!\!*})$
$\exists$	$(A \to iProp) \to iProp$	$((=) \Longrightarrow (\mathrel{-\!\!*}))\ \Longrightarrow (\mathrel{-\!\!*})$

We make use of the Haskell prefix notation, $(\twoheadrightarrow\!\!*)$, to turn an infix operator into a prefix function. The monotonicity of a connective is defined in terms of the requirements we have for each of its arguments and what we know about the resulting statement after application of the arguments. To show how the signature defines monotonicity we will give the definitions of the combinator used to build them.

Question: Bad sentence? But do want to say something about what a signature is.

Definition 3.6: Respectful relation

The respectful relation $R \Longrightarrow R' \colon (A \to B) \to (A \to B) \to \mathit{iProp}$ of two relations $R \colon A \to A \to \mathit{iProp}$, $R' \colon B \to B \to \mathit{iProp}$ is defined as

$$R \Longrightarrow R' \triangleq \lambda f, g.\, \forall x, y.\, R\, x\, y \mathrel{-\!\!*} R'\,(f\, x)\,(g\, y)$$

A signature defines a relation on predicates. It makes use of the two relations, $(\mathrel{-\!\!*})$ and $(=)$. We can now use the signature on the connective

Definition 3.7: Proper element of a relation

Given a relation $R \colon A \to A \to \mathit{iProp}$ and an element $x \in A$, x is a proper element of R if $R\, x\, x$

We define how a connective is monotone by the signature it is a proper element of. The proofs that the connectives are the proper elements of their signature are fairly trivial, but we will highlight the existential qualifier.

Question: I want to expand the first two signatures, but I don't have anything interesting to say about it except for showing the expanded version

We can unfold the definitions in the signature and fill in the existential quantification in order to get the following statement,

$$\forall \Phi, \Psi.\, (\forall x, y.\, x = y \mathrel{-\!\!*} \Phi\, x \mathrel{-\!\!*} \Psi\, y) \mathrel{-\!\!*} (\exists x.\, \Phi\, x) \mathrel{-\!\!*} (\exists x.\, \Psi\, x)$$

This statement can be easily simplified by substituting y for x in the first relation.

$$\forall \Phi, \Psi.\, (\forall x.\, \Phi\, x \mathrel{-\!\!*} \Psi\, x) \mathrel{-\!\!*} (\exists x.\, \Phi\, x) \mathrel{-\!\!*} (\exists x.\, \Psi\, x)$$

We create a new combinator for signatures, the pointwise relation, to include the above simplification in signatures.

Definition 3.8: Pointwise relation

The pointwise relation $\gtrdot R$ is a special case of a respectful relation defined as

$$\gtrdot R \triangleq \lambda f, g.\, \forall x.\, R\,(f\, x)\,(g\, y)$$

The new signature for the existential quantification becomes

$$\gtrdot(\mathrel{-\!\!*}) \Longrightarrow (\mathrel{-\!\!*})$$

Monotone functors To create a monotone functor for the least fixpoint we need to be able to at least define definition 3.1 in terms of the proper element of a signature. We already have most the combinators needed, but we are missing a way to mark a relation as persistent.

Definition 3.9: Persistent relation

The persistent relation $\pmb{\square} R \colon A \to A \to \mathit{iProp}$ for a relation $R \colon A \to A \to \mathit{iProp}$ is defined as

$$\pmb{\square} R \triangleq \lambda x, y.\ \square(R\,x\,y)$$

Thus we can create the following signature for definition 3.1.

$$\pmb{\square}(\gtrdot(-\!\!*)) \Longrightarrow \gtrdot(-\!\!*)$$

Filling in a F as the proper element get the following statement.

$$\square(\forall y.\, \mathit{\Phi}\, y -\!\!* \Psi\, y) -\!\!* \forall x.\, \mathsf{F}\, \mathit{\Phi}\, x -\!\!* \mathsf{F}\, \Psi\, x$$

Which is definition 3.1 but using only wands, instead of entailments. We use the same structure for the signature of $\mathsf{isMLL}_\mathsf{F}$. But we add an extra pointwise to the left and right-hand side of the respectful relation for the extra argument.

$$\pmb{\square}(\gtrdot\gtrdot(-\!\!*)) \Longrightarrow \gtrdot\gtrdot(-\!\!*)$$

We are thus able to write down the monotonicity of a functor using the combinators we have defined.

Monotone proof search To perform the monotone proof search we first have to add one additional lemma.

Lemma 3.10

Any proposition, $P \colon \mathit{iProp}$, is a proper element of the signature $(-\!\!*)$

Proof. Since $(-\!\!*)$ is reflexive, any proposition is immediately a proper element. □

We now show a proof and then outline the steps we took.

Example 3.11: $\mathsf{isMLL}_\mathsf{F}$ is monotone

The predicate $\mathsf{isMLL}_\mathsf{F}$ is monotone in its first argument. Thus, $\mathsf{isMLL}_\mathsf{F}$ is a proper element of

$$\pmb{\square}(\gtrdot\gtrdot(-\!\!*)) \Longrightarrow \gtrdot\gtrdot(-\!\!*)$$

In other words

$$\square\,(\forall hd\,\vec{v}.\,\Phi\,hd\,\vec{v} \mathrel{-\!\!*} \Psi\,hd\,\vec{v}) \mathrel{-\!\!*} \forall hd\,\vec{v}.\,\mathsf{isMLL_F}\,\Phi\,hd\,\vec{v} \mathrel{-\!\!*} \mathsf{isMLL_F}\,\Psi\,hd\,\vec{v}$$

Proof. We assume any premises, $\square\,(\forall hd\,\vec{v}.\,\Phi\,hd\,\vec{v} \mathrel{-\!\!*} \Psi\,hd\,\vec{v})$, and then introduce the universal quantifiers. After unfolding $\mathsf{isMLL_F}$ we have to prove the following.

$$\square\,(\forall hd\,\vec{v}.\,\Phi\,hd\,\vec{v} \mathrel{-\!\!*} \Psi\,hd\,\vec{v}) \vdash (\cdots \lor \cdots \Phi \cdots) \mathrel{-\!\!*} (\cdots \lor \cdots \Psi \cdots)$$

Thus, the top level connective is the wand and the one below it is the disjunction. We now search for a signature ending on a magic wand and which has the disjunction as a proper element. We find the signature $(\mathrel{-\!\!*}) \implies (\mathrel{-\!\!*}) \implies (\mathrel{-\!\!*})$ with $(\lor)$. We apply $((\mathrel{-\!\!*}) \implies (\mathrel{-\!\!*}) \implies (\mathrel{-\!\!*}))(\lor)(\lor)$ resulting in two statements to prove.

$$\square\,(\cdots) \vdash (hd = \textbf{none} * \vec{v} = []) \mathrel{-\!\!*} (hd = \textbf{none} * \vec{v} = [])$$
$$\square\,(\cdots) \vdash (\cdots \Phi \cdots \lor \cdots \Phi \cdots) \mathrel{-\!\!*} (\cdots \Psi \cdots \lor \cdots \Psi \cdots)$$

For the first statement we have as top relation $(\mathrel{-\!\!*})$ with below it $(*)$. We find the signature $(\mathrel{-\!\!*}) \implies (\mathrel{-\!\!*}) \implies (\mathrel{-\!\!*})$ with $(*)$. We apply it and get two new statements. Since both don't have aa almost top level connective we have a signature for we use lemma 3.10 to prove both statements.
The second statement utilizes the same disjunction signature again, thus we just show the end results of applying it.

$$\square\,(\cdots) \vdash (\exists \ell, v', tl.\ \cdots \Phi \cdots) \mathrel{-\!\!*} (\exists \ell, v', tl.\ \cdots \Psi \cdots)$$
$$\square\,(\cdots) \vdash (\exists \ell, v', \vec{v}'', tl.\ \cdots \Phi \cdots) \mathrel{-\!\!*} (\exists \ell, v', \vec{v}'', tl.\ \cdots \Psi \cdots)$$

Both statements have as top level relation $(\mathrel{-\!\!*})$ with below it $\exists$. We apply the signature of $\exists$ with as result.

$$\square\,(\cdots) \vdash \forall \ell.\,(\exists v', tl.\ \cdots \Phi \cdots) \mathrel{-\!\!*} (\exists v', tl.\ \cdots \Psi \cdots)$$
$$\square\,(\cdots) \vdash \forall \ell.\,(\exists v', \vec{v}'', tl.\ \cdots \Phi \cdots) \mathrel{-\!\!*} (\exists v', \vec{v}'', tl.\ \cdots \Psi \cdots)$$

We introduce ℓ and repeat these steps until the existential quantification is no longer tho almost top level connective.

$$\begin{aligned}\square\,(\cdots) \vdash{} & (hd = \textbf{some}\,l * l \mapsto (v', \textbf{true}, tl) * \Phi\,tl\,\vec{v}) \mathrel{-\!\!*} \\ & (hd = \textbf{some}\,l * l \mapsto (v', \textbf{true}, tl) * \Psi\,tl\,\vec{v})\end{aligned}$$
$$\square\,(\cdots) \vdash \begin{pmatrix} hd = \textbf{some}\,l * l \mapsto (v', \textbf{false}, tl) * \\ \vec{v} = v' :: \vec{v}'' * \Phi\,tl\,\vec{v}'' \end{pmatrix} \mathrel{-\!\!*}$$

$$\begin{pmatrix} hd = \mathtt{some}\, l * l \mapsto (v', \mathtt{false}, tl) * \\ \vec{v} = v' :: \vec{v}'' * \Psi\, tl\, \vec{v}'' \end{pmatrix}$$

We can now repeatedly apply the signature of $(*)$ and deal with any created propositions without Φ or Ψ. This leaves us with

$$\Box\, (\forall hd\, \vec{v}.\, \Phi\, hd\, \vec{v} \twoheadrightarrow \Psi\, hd\, \vec{v}) \vdash \Phi\, tl\, \vec{v} \twoheadrightarrow \Psi\, tl\, \vec{v}$$
$$\Box\, (\forall hd\, \vec{v}.\, \Phi\, hd\, \vec{v} \twoheadrightarrow \Psi\, hd\, \vec{v}) \vdash \Phi\, tl\, \vec{v}'' \twoheadrightarrow \Psi\, tl\, \vec{v}''$$

These hold from the assumption. □

Thus, the steps to find a proof are the following. We apply the first step that works.

1. Check if the conclusion follows from the premise, and then apply it.
2. Look for a signature of the almost top level connective where the last relation matches the top level connective of the conclusion. Apply it if we find one. Then introduce any universal quantifiers and modalities.
3. Apply lemma 3.10.

Repeat the above steps for all created branches until it has been proven.