Modal Logic Mechanisation in HOL4

Yiming Xu

15/6/2018

This is a report for summarizing my learning of interactive theorem proving in the HOL. This paper consists of two parts, the first part it about the correspondence theorem on group theory, through the process of searching for methods of proving it, I get known to some basics tools, mainly the simplifiers of the HOL. The second part is about modal logic, where I think is my starting point of being exposed to more advanced and useful tools.

0.1 Correspondence Theorem in the HOL

The main theorem is, stated in usual-sense mathematical language (version on Artin's Algebra):

For a homomorphism $\phi : G \to \mathcal{G}$, call its kernel K. Let H be a subgroup of G that contains K, and let $\mathcal{H}$ be a subgroup of $\mathcal{G}$. Then:
(1) $\phi(H)$ is a subgroup of $\mathcal{G}$.
(2) $\phi^{-1}(\mathcal{H})$ is a subgroup of G.
(3) $\phi^{-1}(\mathcal{H})$ contains K.
(4) $\mathcal{H}$ is a normal subgroup of $\mathcal{G}$ if and only if $\phi^{-1}(\mathcal{H})$ is a normal subgroup of G.
(5) $\phi(\phi^{-1}(\mathcal{H})) = \mathcal{H}$.
(6) $\phi^{-1}(\phi(H)) = H$.
(7) $|\phi^{-1}(\mathcal{H})| = |\mathcal{H}||K|$

In the language of the HOL, the correspondence theorem is stated as:

The $h << g$ denotes the fact that h is a normal subgroup of g. And $h \leq g$ denotes the fact that h is a subgroup of g. All the definitions about group theory are given as functions that take an "object" and returns a boolean to judge whether it satisfies its definition. For instance, *Group g* reads "g is a group".

The right hand side of the arrow is a conjunction, which means that we can split our main goal as smaller subgoals for each conjuncts using `rpt strip_tac` and then prove each subgoal as a lemma. Once we have proved all these lemmas, we can give the remaining work to `metis_tac`.

0.1.1 lemma 1: $\phi(H)$ is a subgroup of $\mathcal{G}$

My Goal:

In the algebra library there are two versions of definition of subgroup, namely `Subgroup` and `subgroup`. Our goal only involves the version `Subgroup`, which it our $\leq$, but one of the theorems in the library which would be used to prove this goal uses the definition `subgroup`. In order to use it, we need to prove that version of definition of subgroup we use implies the other verision. That is:

As this is proved, we can use the existing theorems

The proof would then be strightforward: As $h \leq g1$ then *subgroup h* g_1, then together with the fact *GroupHomo f* g_1 g_2, by the first lemma *GroupHomo f h k*.

And also $h \leq g$ gives *Group* h and *Group* g, then by the last lemma above the goal is proved.

0.1.2 lemma 2: $\phi^{-1}(\mathcal{H})$ is a subgroup of G

Goal: Note the usage of *preimage_group*, which is a function defined as:

It is a function that take a function f, two "groups" g_1, g_2 of types α, β respectively, and a subset h of g_2.

The word "groups" has a quotation mark because it does not denote the group in mathematical sence, instead, it denotes the datatype "group", which consists by three pieces of informations: the underlying set, which it also called the carrier, the identity element, and the operation function defined for every pair of elements in the carrier and returns an element in the carrier.

We cannot encode the information that f is a homomorphism and the gs are groups directly in the definition of *preimage_group*. This is because we are only allowed to define anything on the whole universe of a type. Also the function *preimage_group* does not directly return a group. We only know that the output of it consists of the three pieces of information that a group needs as mentioned above, and to conclude that it is indeed a group when g_1, g_2 are real groups, f is a homomorphism and h is an actual subgroup, we need to prove that the three pieces of information defined by this sort of inputs of *preimage_group* satisfies the group axioms.

To prove this goal, we mainly want to prove that the preimage of a subgroup of a group under a homomorphism is a group. Once we prove

then the main goal then easily followes from here.

0.1.3 lemma 3: $K \subseteq \phi^{-1}(\mathcal{H})$

0.2 Modal logic in the HOL

My Script files on modal logic basically follow the book of Blackburn's book "Modal Logic".

0.2.1 Chapter 1

Basic Constructions

Everything discussed in this report is with respect to the following modal language-called the basic modal language. In this language, a formulas of a type α are:

$$\alpha\ \phi\ =\ \mathsf{VAR}\ \alpha\ |\ \mathsf{DISJ}\ (\alpha\ \phi)\ (\alpha\ \phi)\ |\ \bot\ |\ (\neg)\ (\alpha\ \phi)\ |\ \Diamond\ (\alpha\ \phi)$$

The $\Box, \rightarrow, \wedge, \leftrightarrow$ and the tautology are defined as:

$$\Box\ \phi\ \stackrel{\mathsf{def}}{=}\ \neg\Diamond\ (\neg\phi)$$

$$f_1 \ \rightarrow \ f_2 \ \stackrel{\text{def}}{=} \ \mathsf{DISJ}\ (\neg f_1)\ f_2$$

$$\mathsf{AND}\ f_1\ f_2 \ \stackrel{\text{def}}{=} \ \neg\mathsf{DISJ}\ (\neg f_1)\ (\neg f_2)$$

$$\mathsf{DOUBLE\_IMP}\ f_1\ f_2 \ \stackrel{\text{def}}{=} \ \mathsf{AND}\ (f_1 \ \rightarrow \ f_2)\ (f_2 \ \rightarrow \ f_1)$$

$$\mathsf{TRUE} \ \stackrel{\text{def}}{=} \ \neg\bot$$

respectively.

We can define any function that sends each element in the universal set of type α to a β-formula, such a function f induces a map from the collection of all the α-formulas to the collections of β-formulas. It is called the substitution map induced by the function f, it is defined as:

$$\begin{array}{l}
\mathsf{subst}\ f\ \bot \ \stackrel{\text{def}}{=} \ \bot \\
\mathsf{subst}\ f\ (\mathsf{VAR}\ p) \ \stackrel{\text{def}}{=} \ f\ p \\
\mathsf{subst}\ f\ (\mathsf{DISJ}\ \phi_1\ \phi_2) \ \stackrel{\text{def}}{=} \ \mathsf{DISJ}\ (\mathsf{subst}\ f\ \phi_1)\ (\mathsf{subst}\ f\ \phi_2) \\
\mathsf{subst}\ f\ (\neg\phi) \ \stackrel{\text{def}}{=} \ \neg\mathsf{subst}\ f\ \phi \\
\mathsf{subst}\ f\ (\Diamond\ \phi) \ \stackrel{\text{def}}{=} \ \Diamond\ (\mathsf{subst}\ f\ \phi)
\end{array}$$

Directly from its definition, the substitution rule on a formula involved the $\Box$ is given by:

$$\vdash \mathsf{subst}\ f\ (\Box\ \phi) \ = \ \Box\ (\mathsf{subst}\ f\ \phi)$$

We would study formulas in basic modal language on models. A model of type α consists of three pieces of informations: a set of worlds, a relation defined on the set of worlds, and a evaluation map from the set of propositional letters of type α to the power set of the set of worlds. The first two pieces together is called a frame. It is created as a type(for further convenience, we only allow the worlds in a model to be natural numbers):

$$\alpha\ \texttt{frame} = \texttt{<|}\ \mathsf{world}\ :\ \alpha\ \rightarrow\ \texttt{bool};\ \mathsf{rel}\ :\ \alpha\ \rightarrow\ \alpha\ \rightarrow\ \texttt{bool}\ \texttt{|>}$$

And a model is defined as a type based on a frame, it is:

$$\begin{array}{l}
(\alpha,\ \beta)\ \texttt{model} = \texttt{<|} \\
\quad \mathsf{frame}\ :\ \beta\ \texttt{frame}; \\
\quad \mathsf{valt}\ :\ \alpha\ \rightarrow\ \beta\ \rightarrow\ \texttt{bool} \\
\texttt{|>}
\end{array}$$

A large part of study of formulas of basic modal language is about modal satisfication. Satisfication is defined on the elements of the world set of a model, so we need to ensure that once we say "a formula is satisfied at a world of a

model", we must ensure that the world we are talking about is actually in model. So the definition of satisfication is encoded as:

$$\begin{array}{l}
\mathsf{satis}\ \mathfrak{M}\ w\ (\mathsf{VAR}\ p) \stackrel{\mathsf{def}}{=} w \in \mathfrak{M}.\mathsf{frame.world} \wedge w \in \mathfrak{M}.\mathsf{valt}\ p \\
\mathsf{satis}\ \mathfrak{M}\ w\ \bot \stackrel{\mathsf{def}}{=} w \in \mathfrak{M}.\mathsf{frame.world} \wedge \mathsf{F} \\
\mathsf{satis}\ \mathfrak{M}\ w\ (\neg\phi) \stackrel{\mathsf{def}}{=} w \in \mathfrak{M}.\mathsf{frame.world} \wedge \neg\mathsf{satis}\ \mathfrak{M}\ w\ \phi \\
\mathsf{satis}\ \mathfrak{M}\ w\ (\mathsf{DISJ}\ \phi_1\ \phi_2) \stackrel{\mathsf{def}}{=} \mathsf{satis}\ \mathfrak{M}\ w\ \phi_1 \vee \mathsf{satis}\ \mathfrak{M}\ w\ \phi_2 \\
\mathsf{satis}\ \mathfrak{M}\ w\ (\Diamond\ \phi) \stackrel{\mathsf{def}}{=} \\
\quad w \in \mathfrak{M}.\mathsf{frame.world} \wedge \\
\quad \exists\, v.\ \mathfrak{M}.\mathsf{frame.rel}\ w\ v \wedge v \in \mathfrak{M}.\mathsf{frame.world} \wedge \mathsf{satis}\ \mathfrak{M}\ v\ \phi
\end{array}$$

Thus latter if we want to to prove a formula is satisfied at some point in the model, firstly we need the fact that the point is indeed in the world set of the model.

Instead of regarding a single formula, we can also say a set of formulas of some fixed type is satisfied at a world. We say a set of formulas is satisfied at a point of a model when each formula in the set is satisfied in at this point. In the HOL, it looks like:

$$\mathsf{satis\_set}\ \mathfrak{M}\ w\ \Sigma \stackrel{\mathsf{def}}{=} \forall\, a.\ a \in \Sigma \Rightarrow \mathsf{satis}\ \mathfrak{M}\ w\ a$$

Regarding a formula and its satisfiability on a given model, it is universally true if it is satisfied at every world in the model, it is satisfiable if there exists some worlds in the model where it is satisfied, and it is refutable if there exists some worlds in the model where its negation is satisfied, just as:

$$\mathsf{stfable}\ \mathfrak{M}\ \phi \stackrel{\mathsf{def}}{=} \exists\, w.\ \mathsf{satis}\ \mathfrak{M}\ w\ \phi$$

$$\mathsf{refutable}\ \mathfrak{M}\ \phi \stackrel{\mathsf{def}}{=} \exists\, w.\ \mathsf{satis}\ \mathfrak{M}\ w\ (\neg\phi)$$

We use a bunch of definition regarding "validness" to describe the satisfiability of a formula over a frame. In fact, the validness of a formula can be defined on the level of a state of a frame, a frame or a class of frames. A formula ϕ is valid at a state w in a frame if for any model over this frame, ϕ is satisfied at w. ϕ is valid in a frame if it is valid at every state in the frame, and is valid in a class C of frames if for any frame f in C, ϕ is valid in f. Finally, to say a formula is valid is to say that for any frame f, it is valid in f. These definitions are expressed as follows:

$$\mathsf{valid\_frame\_state}\ f\ w\ \phi \stackrel{\mathsf{def}}{=} \forall\, \mathfrak{M}.\ \mathfrak{M}.\mathsf{frame} = f \Rightarrow \mathsf{satis}\ \mathfrak{M}\ w\ \phi$$

$$\mathsf{valid\_frame}\ f\ \phi \stackrel{\mathsf{def}}{=} \forall\, w.\ \mathsf{valid\_frame\_state}\ f\ w\ \phi$$

$$\mathsf{valid\_class}\ C\ \phi \stackrel{\mathsf{def}}{=} \forall f.\ f \in C \Rightarrow \mathsf{valid\_frame}\ f\ \phi$$

$$\vdash \mathsf{valid}\ \phi \iff \forall f.\ \mathsf{valid\_frame}\ f\ \phi$$

For a class of frames, the "logic" of it is the collection of all formulas that are valid in every frames in this class:

$$\mathsf{LOGIC}\ C \stackrel{\mathsf{def}}{=} \{\ \phi \mid \mathsf{valid\_class}\ C\ \phi\ \}$$

For a set S of α-models and a set Σ for α-formulas, saying an α-formula ϕ is a local semantic consequence of Σ over S is saying that for any model M in S, and any point w in M, if Σ is satisfied, then ϕ must be satisfied. Stated in the language of HOL, it is:

$$\mathsf{LSC}\ \Sigma\ S\ \phi \stackrel{\mathsf{def}}{=} \forall \mathfrak{M}\ w.\ \mathfrak{M} \in S \wedge \mathsf{satis\_set}\ \mathfrak{M}\ w\ \Sigma \Rightarrow \mathsf{satis}\ \mathfrak{M}\ w\ \phi$$

So far what we have already defined are all about semantics, and in fact, there is a syntactic mechanism of obtaining the collection of valid formulas. This syntactic mechanism involves the usage of K-proof. In usual language, it is defined as:

A K-proof is a finite sequence of formulas, each of which is an axiom, or follows from one or more earlier items in the sequence by applying a rule of proof.

For the axioms, they are all propositional tautologies plus:
$\Box(p \to q) \to (\Box p \to \Box q)$
$\Diamond p \leftrightarrow \neg\Box\neg p$

And the rules are: (1) Modens ponens: if ϕ and $\phi \to \psi$ both occurs as lines of some line of a K-proof, then we can append a line ψ so the resulting list is also a K-proof.
(2) Uniform substitution: if ϕ occurs as a line of a K-proof, then we can append a line θ where θ is obtained from ϕ by applying some substitution function.
(3) Generalization: if ϕ occurs as a line of a K-proof, then we can append a line $\Box\phi$ to the original list and obtain a K-proof.

Given the rules of the K-proof, we can know about three things: the rule itself which defines what sort of list is called a K-proof, a induction principle according to the rule which allows us to use in proving things about Kproofs by induction, and if a given list is a K-proof, what are the possible cases of its form. So we can define these three things altogether, with the application of `Hol_reln`:

It gives three statements at once, namely:

$$
\begin{array}{l}
\vdash \mathsf{Kproof}\ [] \ \wedge \\
\quad (\forall\, p\ \phi_1\ \phi_2. \\
\qquad \mathsf{Kproof}\ p \ \wedge\ \mathsf{MEM}\ (\phi_1 \ \rightarrow\ \phi_2)\ p \ \wedge\ \mathsf{MEM}\ \phi_1\ p \ \Rightarrow \\
\qquad \mathsf{Kproof}\ (p \ +\!\!+\ [\phi_2])) \ \wedge \\
\quad (\forall\, p\ \phi\ f.\ \mathsf{Kproof}\ p \ \wedge\ \mathsf{MEM}\ \phi\ p \ \Rightarrow\ \mathsf{Kproof}\ (p \ +\!\!+\ [\mathsf{subst}\ f\ \phi])) \ \wedge \\
\quad (\forall\, p\ \phi.\ \mathsf{Kproof}\ p \ \wedge\ \mathsf{MEM}\ \phi\ p \ \Rightarrow\ \mathsf{Kproof}\ (p \ +\!\!+\ [\Box\, \phi])) \ \wedge \\
\quad (\forall\, p\ \phi_1\ \phi_2. \\
\qquad \mathsf{Kproof}\ p \ \Rightarrow\ \mathsf{Kproof}\ (p \ +\!\!+\ [\Box\, (\phi_1 \ \rightarrow\ \phi_2) \ \rightarrow\ \Box\, \phi_1 \ \rightarrow\ \Box\, \phi_2])) \ \wedge \\
\quad (\forall\, p\ \phi.\ \mathsf{Kproof}\ p \ \Rightarrow\ \mathsf{Kproof}\ (p \ +\!\!+\ [\Diamond\, \phi \ \rightarrow\ \neg\Box\, (\neg\phi)])) \ \wedge \\
\quad (\forall\, p\ \phi.\ \mathsf{Kproof}\ p \ \Rightarrow\ \mathsf{Kproof}\ (p \ +\!\!+\ [\neg\Box\, (\neg\phi) \ \rightarrow\ \Diamond\, \phi])) \ \wedge \\
\quad \forall\, p\ f.\ \mathsf{Kproof}\ p \ \wedge\ \mathsf{ptaut}\ f \ \Rightarrow\ \mathsf{Kproof}\ (p \ +\!\!+\ [f])
\end{array}
$$

$$
\begin{array}{l}
\vdash \mathit{Kproof'}\ [] \ \wedge \\
\quad (\forall\, p\ \phi_1\ \phi_2. \\
\qquad \mathit{Kproof'}\ p \ \wedge\ \mathsf{MEM}\ (\phi_1 \ \rightarrow\ \phi_2)\ p \ \wedge\ \mathsf{MEM}\ \phi_1\ p \ \Rightarrow \\
\qquad \mathit{Kproof'}\ (p \ +\!\!+\ [\phi_2])) \ \wedge \\
\quad (\forall\, p\ \phi\ f.\ \mathit{Kproof'}\ p \ \wedge\ \mathsf{MEM}\ \phi\ p \ \Rightarrow\ \mathit{Kproof'}\ (p \ +\!\!+\ [\mathsf{subst}\ f\ \phi])) \ \wedge \\
\quad (\forall\, p\ \phi.\ \mathit{Kproof'}\ p \ \wedge\ \mathsf{MEM}\ \phi\ p \ \Rightarrow\ \mathit{Kproof'}\ (p \ +\!\!+\ [\Box\, \phi])) \ \wedge \\
\quad (\forall\, p\ \phi_1\ \phi_2. \\
\qquad \mathit{Kproof'}\ p \ \Rightarrow\ \mathit{Kproof'}\ (p \ +\!\!+\ [\Box\, (\phi_1 \ \rightarrow\ \phi_2) \ \rightarrow\ \Box\, \phi_1 \ \rightarrow\ \Box\, \phi_2])) \ \wedge \\
\quad (\forall\, p\ \phi.\ \mathit{Kproof'}\ p \ \Rightarrow\ \mathit{Kproof'}\ (p \ +\!\!+\ [\Diamond\, \phi \ \rightarrow\ \neg\Box\, (\neg\phi)])) \ \wedge \\
\quad (\forall\, p\ \phi.\ \mathit{Kproof'}\ p \ \Rightarrow\ \mathit{Kproof'}\ (p \ +\!\!+\ [\neg\Box\, (\neg\phi) \ \rightarrow\ \Diamond\, \phi])) \ \wedge \\
\quad (\forall\, p\ f.\ \mathit{Kproof'}\ p \ \wedge\ \mathsf{ptaut}\ f \ \Rightarrow\ \mathit{Kproof'}\ (p \ +\!\!+\ [f])) \ \Rightarrow \\
\quad \forall\, a_0.\ \mathsf{Kproof}\ a_0 \ \Rightarrow\ \mathit{Kproof'}\ a_0
\end{array}
$$

$$
\begin{array}{l}
\vdash \mathsf{Kproof}\ a_0 \ \Longleftrightarrow \\
\quad a_0 \ =\ [] \ \vee \\
\quad (\exists\, p\ \phi_1\ \phi_2. \\
\qquad a_0 \ =\ p \ +\!\!+\ [\phi_2] \ \wedge\ \mathsf{Kproof}\ p \ \wedge\ \mathsf{MEM}\ (\phi_1 \ \rightarrow\ \phi_2)\ p \ \wedge \\
\qquad \mathsf{MEM}\ \phi_1\ p) \ \vee \\
\quad (\exists\, p\ \phi\ f.\ a_0 \ =\ p \ +\!\!+\ [\mathsf{subst}\ f\ \phi] \ \wedge\ \mathsf{Kproof}\ p \ \wedge\ \mathsf{MEM}\ \phi\ p) \ \vee \\
\quad (\exists\, p\ \phi.\ a_0 \ =\ p \ +\!\!+\ [\Box\, \phi] \ \wedge\ \mathsf{Kproof}\ p \ \wedge\ \mathsf{MEM}\ \phi\ p) \ \vee \\
\quad (\exists\, p\ \phi_1\ \phi_2. \\
\qquad a_0 \ =\ p \ +\!\!+\ [\Box\, (\phi_1 \ \rightarrow\ \phi_2) \ \rightarrow\ \Box\, \phi_1 \ \rightarrow\ \Box\, \phi_2] \ \wedge\ \mathsf{Kproof}\ p) \ \vee \\
\quad (\exists\, p\ \phi.\ a_0 \ =\ p \ +\!\!+\ [\Diamond\, \phi \ \rightarrow\ \neg\Box\, (\neg\phi)] \ \wedge\ \mathsf{Kproof}\ p) \ \vee \\
\quad (\exists\, p\ \phi.\ a_0 \ =\ p \ +\!\!+\ [\neg\Box\, (\neg\phi) \ \rightarrow\ \Diamond\, \phi] \ \wedge\ \mathsf{Kproof}\ p) \ \vee \\
\quad \exists\, p\ f.\ a_0 \ =\ p \ +\!\!+\ [f] \ \wedge\ \mathsf{Kproof}\ p \ \wedge\ \mathsf{ptaut}\ f
\end{array}
$$

The significance of the K-proof system can be seen through this fact: A basic modal formula is K-provable if and only if it is valid.

The definition of normal modal logic is a direct abstraction from the definition of the K-proof system. It is defined as a set of formulas closed under modens ponens, uniform substitution and generalization, Which is expressed as:

$$\begin{aligned}
&\mathsf{NML}\ S \stackrel{\mathsf{def}}{=} \\
&\ \forall A\ B\ p\ q\ f\ \phi. \\
&\quad (\mathsf{ptaut}\ \phi \Rightarrow \phi \in S) \wedge (\Box(p \to q) \to \Box p \to \Box q) \in S \wedge \\
&\quad (\Diamond p \to \neg\Box(\neg p)) \in S \wedge (\neg\Box(\neg p) \to \Diamond p) \in S \wedge \\
&\quad (A \in S \Rightarrow \mathsf{subst}\ f\ A \in S) \wedge (A \in S \Rightarrow \Box A \in S) \wedge \\
&\quad ((A \to B) \in S \wedge A \in S \Rightarrow B \in S)
\end{aligned}$$

Exercises

There are two exercises in chapter 1.6 which have been proved in my file.

Exercise 1.6.2 Goal: Let ϕ^- be the 'demodalized' version of a modal formula ϕ, that is, ϕ^- is obtained from ϕ by erasing all the diamonds. Then if ϕ is K-provable, then ϕ^- is a propositional tautology.

To solve this exercise, firstly we need to define the demodalization for a formula. Demodalizing is the operation that for diamonded formulas, removing the diamond, and extending to all the formulas, that is(quite similar to the definition of substitution map):

$$\begin{aligned}
&\mathsf{demodalize}\ \bot \stackrel{\mathsf{def}}{=} \bot \\
&\mathsf{demodalize}\ (\mathsf{VAR}\ p) \stackrel{\mathsf{def}}{=} \mathsf{VAR}\ p \\
&\mathsf{demodalize}\ (\mathsf{DISJ}\ \phi_1\ \phi_2) \stackrel{\mathsf{def}}{=} \mathsf{DISJ}\ (\mathsf{demodalize}\ \phi_1)\ (\mathsf{demodalize}\ \phi_2) \\
&\mathsf{demodalize}\ (\neg\phi) \stackrel{\mathsf{def}}{=} \neg\mathsf{demodalize}\ \phi \\
&\mathsf{demodalize}\ (\Diamond\phi) \stackrel{\mathsf{def}}{=} \mathsf{demodalize}\ \phi
\end{aligned}$$

Secondly, we need to state clearly which kind of formula is a propositional tautology. To do this, firstly we need to define what is a propositional formula, then the definition of propositional tautology would be given by: for a formula ϕ, ϕ is a propositional tautology if and only if ϕ is a propositional tautology, and ϕ 'always holds'.

A propositional formula is just a modal formula without the diamond, so it can be defined as:

$$\begin{aligned}
&\mathsf{propform}\ (\mathsf{VAR}\ p) \stackrel{\mathsf{def}}{=} \mathsf{T} \\
&\mathsf{propform}\ (\mathsf{DISJ}\ \phi_1\ \phi_2) \stackrel{\mathsf{def}}{=} \mathsf{propform}\ \phi_1 \wedge \mathsf{propform}\ \phi_2 \\
&\mathsf{propform}\ (\neg f) \stackrel{\mathsf{def}}{=} \mathsf{propform}\ f \\
&\mathsf{propform}\ \bot \stackrel{\mathsf{def}}{=} \mathsf{T} \\
&\mathsf{propform}\ (\Diamond f) \stackrel{\mathsf{def}}{=} \mathsf{F}
\end{aligned}$$

Similar as mentioned in section on the correspondence theorem. propform is a function that takes a formula and returns a truth-value, that is, it judges whether a modal formula is a propositional formula.

Then we will need to describe what is 'always holds' for a propositional formula. Here we describe this property as: for any assignment of truth values

of propositional symbols involved in this formula, the truth value of the whole formula must be T. Here we need to define a evaluation function that can give us a truth value of a propositional formula for each assignment of the propositional symbols. it is defined as:

$$\begin{aligned}
&\mathsf{peval}\ \sigma\ (\mathsf{VAR}\ p) \stackrel{\mathsf{def}}{=} \sigma\ p \\
&\mathsf{peval}\ \sigma\ (\mathsf{DISJ}\ f_1\ f_2) \stackrel{\mathsf{def}}{=} \mathsf{peval}\ \sigma\ f_1\ \vee\ \mathsf{peval}\ \sigma\ f_2 \\
&\mathsf{peval}\ \sigma\ \bot \stackrel{\mathsf{def}}{=} \mathsf{F} \\
&\mathsf{peval}\ \sigma\ (\neg f) \stackrel{\mathsf{def}}{=} \neg\mathsf{peval}\ \sigma\ f \\
&\mathsf{peval}\ \sigma\ (\Diamond\ f) \stackrel{\mathsf{def}}{=} \mathsf{F}
\end{aligned}$$

Here σ is an assignment of truth-values on the propositional symbols, it is a function sending each propositional symbol to its assigned truth-value.

Thus as explained above, we can define propositional tautology as:

$$\mathsf{ptaut}\ f \stackrel{\mathsf{def}}{=} \mathsf{propform}\ f\ \wedge\ \forall\sigma.\ \mathsf{peval}\ \sigma\ f\ \iff\ \mathsf{T}$$

Have these definitions constructed, now we are aim to prove that once a formula ϕ is K-provable, means that it occurs at the bottom line of a K-proof, then ϕ^- is a propostional tautology. Noting that almost everything here is defined inductively, so this goal is suitable to be prove by induction on 'K-proof'. The trick is a stronger goal gives a stronger inductive hypothesis, which can be more useful. Here to make the inductive hypothesis more useful, we strongthen the goal as:

$$\vdash \mathsf{Kproof}\ p\ \Rightarrow\ \forall f.\ \mathsf{MEM}\ f\ p\ \Rightarrow\ \mathsf{ptaut}\ (\mathsf{demodalize}\ f)$$

That is, for every line in a K-proof, it can be demodalized to give a propositional tautology.

The induction gives subgoals looks like:

$\mathsf{ptaut}\ (\mathsf{demodalize}\ f)$

0. $\mathsf{Kproof}\ p$
1. $\forall f.\ \mathsf{MEM}\ f\ p\ \Rightarrow\ \mathsf{ptaut}\ (\mathsf{demodalize}\ f)$
2. $\mathsf{MEM}\ (\phi_1\ \rightarrow\ \phi_2)\ p$
3. $\mathsf{MEM}\ \phi_1\ p$
4. $\mathsf{MEM}\ f\ (p\ +\!\!+\ [\phi_2])$

The line above the solid line is the goal, and the lines under the line are the assumptions. To prove it, observing that by assumption 4, either f in p or $f\ =\ \phi_2$. For the first case, the inductive hypothesis, which is assumption 1, applies. For the second case, by assumption 3 and the inductive hypothesis, $\phi_1\ \rightarrow\ \phi_2$ is a propositional tautology, and then we can prove

$$\vdash \mathsf{ptaut}\ (f_1\ \rightarrow\ f_2)\ \wedge\ \mathsf{ptaut}\ f_1\ \Rightarrow\ \mathsf{ptaut}\ f_2$$

from definitions "peval" and "ptaut".

Then by the lemma above, the subgoal is solved. All the subgoals except the following one can be solved in a similar way.

The trickest subgoal is:

$$\frac{\mathsf{ptaut}\ (\mathsf{demodalize}\ f')}{\begin{array}{l}0.\ \mathsf{Kproof}\ p\\ 1.\ \forall f.\ \mathsf{MEM}\ f\ p\ \Rightarrow\ \mathsf{ptaut}\ (\mathsf{demodalize}\ f)\\ 2.\ \mathsf{MEM}\ \phi\ p\\ 3.\ \mathsf{MEM}\ f'\ (p\ +\!\!+\ [\mathsf{subst}\ f\ \phi])\end{array}}$$

As before, f' is either in p or is $\mathsf{subst}\ f\ \phi$. The former case is trivial by assumption 1. And for the later case, we aim to prove that $\mathsf{demodalize}\ (\mathsf{subst}\ f\ \phi)$ is a propositional tautology, means that $\forall\sigma.\ \mathsf{peval}\ \sigma\ (\mathsf{demodalize}\ (\mathsf{subst}\ f\ \phi)) \iff \mathsf{T}$. As as ϕ is in the list p, by the inductive hypothesis, $\mathsf{demodalize}\ \phi$ is a tautology. So once we prove that $\forall\sigma$.
$\exists\sigma'$.
pevel σ demodalize $(\mathsf{subst}\ f\ \phi) \iff$
$\mathsf{peval}\ \sigma'\ (\mathsf{demodalize}\ \phi)$, as the right hand side has truth value T, then we are done.

So we prove:

$$\begin{array}{l}\vdash \mathsf{propform}\ \phi\ \Rightarrow\\ \quad (\mathsf{peval}\ \sigma\ (\mathsf{demodalize}\ (\mathsf{subst}\ f\ \phi)) \iff\\ \quad\quad \mathsf{peval}\ (\mathsf{peval}\ \sigma\ \circ\ \mathsf{demodalize}\ \circ\ f)\ \phi)\end{array}$$

by inducting on ϕ and using the definition of demodalization.

Note that $\mathsf{demodalize}\ f$ is a propositional formula so the above lemma applies, and after proving:

$$\begin{array}{l}\vdash \mathsf{demodalize}\ (\mathsf{subst}\ f\ \phi)\ =\\ \quad \mathsf{demodalize}\ (\mathsf{subst}\ f\ (\mathsf{demodalize}\ \phi))\end{array}$$

We can conclude $\mathsf{peval}\ \sigma\ (\mathsf{demodalize}\ (\mathsf{subst}\ f\ (\mathsf{demodalize}\ \phi))) \iff$
$\mathsf{peval}\ \sigma\ (\mathsf{demodalize}\ (\mathsf{subst}\ f\ \phi)) = \mathsf{peval}\ (\mathsf{peval}\ \sigma\ \circ\ \mathsf{demodalize}\ \circ\ f)\ \phi \iff \mathsf{T}$.

Exercise 1.6.6 Goal: Given a set of formulas Γ. Suppose we form the axiom system $K\Gamma$ by adding as axioms all the formulas in Γ to K. Then the Hilbert system $K\Gamma$ proves precisely the formulas contained in the normal modal logic $K\Gamma$.

Firstly, we setup the Hilbert system KΓ by defining the rule of proving things in $K\Gamma$. Almost the same as the definition of K-proof, we use `Hol_reln` to give:

$$
\begin{array}{l}
\vdash \mathsf{KGproof}\ \Gamma\ []\ \wedge \\
\quad (\forall\, p\ \phi_1\ \phi_2. \\
\qquad \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \mathsf{MEM}\ (\phi_1\ \rightarrow\ \phi_2)\ p\ \wedge\ \mathsf{MEM}\ \phi_1\ p\ \Rightarrow \\
\qquad \mathsf{KGproof}\ \Gamma\ (p\ +\!\!+\ [\phi_2]))\ \wedge \\
\quad (\forall\, p\ \phi\ f. \\
\qquad \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \mathsf{MEM}\ \phi\ p\ \Rightarrow\ \mathsf{KGproof}\ \Gamma\ (p\ +\!\!+\ [\mathsf{subst}\ f\ \phi]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \mathsf{MEM}\ \phi\ p\ \Rightarrow\ \mathsf{KGproof}\ \Gamma\ (p\ +\!\!+\ [\Box\,\phi]))\ \wedge \\
\quad (\forall\, p\ \phi_1\ \phi_2. \\
\qquad \mathsf{KGproof}\ \Gamma\ p\ \Rightarrow \\
\qquad \mathsf{KGproof}\ \Gamma\ (p\ +\!\!+\ [\Box\,(\phi_1\ \rightarrow\ \phi_2)\ \rightarrow\ \Box\,\phi_1\ \rightarrow\ \Box\,\phi_2]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathsf{KGproof}\ \Gamma\ p\ \Rightarrow\ \mathsf{KGproof}\ \Gamma\ (p\ +\!\!+\ [\Diamond\,\phi\ \rightarrow\ \neg\Box\,(\neg\phi)]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathsf{KGproof}\ \Gamma\ p\ \Rightarrow\ \mathsf{KGproof}\ \Gamma\ (p\ +\!\!+\ [\neg\Box\,(\neg\phi)\ \rightarrow\ \Diamond\,\phi]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \mathsf{ptaut}\ \phi\ \Rightarrow\ \mathsf{KGproof}\ \Gamma\ (p\ +\!\!+\ [\phi]))\ \wedge \\
\quad \forall\, p\ \phi.\ \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \phi\ \in\ \Gamma\ \Rightarrow\ \mathsf{KGproof}\ \Gamma\ (p\ +\!\!+\ [\phi])
\end{array}
$$

$$
\begin{array}{l}
\vdash \mathit{KGproof'}\ []\ \wedge \\
\quad (\forall\, p\ \phi_1\ \phi_2. \\
\qquad \mathit{KGproof'}\ p\ \wedge\ \mathsf{MEM}\ (\phi_1\ \rightarrow\ \phi_2)\ p\ \wedge\ \mathsf{MEM}\ \phi_1\ p\ \Rightarrow \\
\qquad \mathit{KGproof'}\ (p\ +\!\!+\ [\phi_2]))\ \wedge \\
\quad (\forall\, p\ \phi\ f.\ \mathit{KGproof'}\ p\ \wedge\ \mathsf{MEM}\ \phi\ p\ \Rightarrow\ \mathit{KGproof'}\ (p\ +\!\!+\ [\mathsf{subst}\ f\ \phi]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathit{KGproof'}\ p\ \wedge\ \mathsf{MEM}\ \phi\ p\ \Rightarrow\ \mathit{KGproof'}\ (p\ +\!\!+\ [\Box\,\phi]))\ \wedge \\
\quad (\forall\, p\ \phi_1\ \phi_2. \\
\qquad \mathit{KGproof'}\ p\ \Rightarrow \\
\qquad \mathit{KGproof'}\ (p\ +\!\!+\ [\Box\,(\phi_1\ \rightarrow\ \phi_2)\ \rightarrow\ \Box\,\phi_1\ \rightarrow\ \Box\,\phi_2]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathit{KGproof'}\ p\ \Rightarrow\ \mathit{KGproof'}\ (p\ +\!\!+\ [\Diamond\,\phi\ \rightarrow\ \neg\Box\,(\neg\phi)]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathit{KGproof'}\ p\ \Rightarrow\ \mathit{KGproof'}\ (p\ +\!\!+\ [\neg\Box\,(\neg\phi)\ \rightarrow\ \Diamond\,\phi]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathit{KGproof'}\ p\ \wedge\ \mathsf{ptaut}\ \phi\ \Rightarrow\ \mathit{KGproof'}\ (p\ +\!\!+\ [\phi]))\ \wedge \\
\quad (\forall\, p\ \phi.\ \mathit{KGproof'}\ p\ \wedge\ \phi\ \in\ \Gamma\ \Rightarrow\ \mathit{KGproof'}\ (p\ +\!\!+\ [\phi]))\ \Rightarrow \\
\quad \forall\, a_0.\ \mathsf{KGproof}\ \Gamma\ a_0\ \Rightarrow\ \mathit{KGproof'}\ a_0
\end{array}
$$

$$
\begin{array}{l}
\vdash \mathsf{KGproof}\ \Gamma\ a_0\ \Longleftrightarrow \\
\quad a_0\ =\ []\ \vee \\
\quad (\exists\, p\ \phi_1\ \phi_2. \\
\qquad a_0\ =\ p\ +\!\!+\ [\phi_2]\ \wedge\ \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \mathsf{MEM}\ (\phi_1\ \rightarrow\ \phi_2)\ p\ \wedge \\
\qquad \mathsf{MEM}\ \phi_1\ p)\ \vee \\
\quad (\exists\, p\ \phi\ f.\ a_0\ =\ p\ +\!\!+\ [\mathsf{subst}\ f\ \phi]\ \wedge\ \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \mathsf{MEM}\ \phi\ p)\ \vee \\
\quad (\exists\, p\ \phi.\ a_0\ =\ p\ +\!\!+\ [\Box\,\phi]\ \wedge\ \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \mathsf{MEM}\ \phi\ p)\ \vee \\
\quad (\exists\, p\ \phi_1\ \phi_2. \\
\qquad a_0\ =\ p\ +\!\!+\ [\Box\,(\phi_1\ \rightarrow\ \phi_2)\ \rightarrow\ \Box\,\phi_1\ \rightarrow\ \Box\,\phi_2]\ \wedge \\
\qquad \mathsf{KGproof}\ \Gamma\ p)\ \vee \\
\quad (\exists\, p\ \phi.\ a_0\ =\ p\ +\!\!+\ [\Diamond\,\phi\ \rightarrow\ \neg\Box\,(\neg\phi)]\ \wedge\ \mathsf{KGproof}\ \Gamma\ p)\ \vee \\
\quad (\exists\, p\ \phi.\ a_0\ =\ p\ +\!\!+\ [\neg\Box\,(\neg\phi)\ \rightarrow\ \Diamond\,\phi]\ \wedge\ \mathsf{KGproof}\ \Gamma\ p)\ \vee \\
\quad (\exists\, p\ \phi.\ a_0\ =\ p\ +\!\!+\ [\phi]\ \wedge\ \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \mathsf{ptaut}\ \phi)\ \vee \\
\quad \exists\, p\ \phi.\ a_0\ =\ p\ +\!\!+\ [\phi]\ \wedge\ \mathsf{KGproof}\ \Gamma\ p\ \wedge\ \phi\ \in\ \Gamma
\end{array}
$$

Also similar to the mentioned definition of normal modal logic, we define the normal modal logic KΓ as:

$$\mathsf{NMLG}\ \Gamma \stackrel{\mathsf{def}}{=} \mathsf{BIGINTER}\ \{\ A\ |\ \mathsf{NML}\ A\ \wedge\ \Gamma\ \subseteq\ A\ \}$$

$\mathsf{NMLG}\ \Gamma$ is the smallest normal modal logic containing Γ. Once we mention the word "smallest", we know that there is a induction principle here, we can derive this principle as:

$$\begin{array}{l}\vdash (\forall A\ B\ p\ q\ f\ \phi.\\ \quad (\mathsf{ptaut}\ \phi\ \Rightarrow\ \phi\ \in\ P)\ \wedge\ (\Box\ (p\ \rightarrow\ q)\ \rightarrow\ \Box\ p\ \rightarrow\ \Box\ q)\ \in\ P\ \wedge\\ \quad (\Diamond\ p\ \rightarrow\ \neg\Box\ (\neg p))\ \in\ P\ \wedge\ (\neg\Box\ (\neg p)\ \rightarrow\ \Diamond\ p)\ \in\ P\ \wedge\\ \quad (A\ \in\ P\ \Rightarrow\ \mathsf{subst}\ f\ A\ \in\ P)\ \wedge\ (A\ \in\ P\ \Rightarrow\ \Box\ A\ \in\ P)\ \wedge\\ \quad ((A\ \rightarrow\ B)\ \in\ P\ \wedge\ A\ \in\ P\ \Rightarrow\ B\ \in\ P))\ \wedge\ G\ \subseteq\ P\ \Rightarrow\\ \forall \phi.\ \phi\ \in\ \mathsf{NMLG}\ G\ \Rightarrow\ \phi\ \in\ P\end{array}$$

This principle is used to derive a conclusion that once a formula is in the normal modal logic, the predicate P holds for this formula. In one direction, we are proving that all the formulas in the normal modal logic is $K\Gamma$-provable. The goal looks like:

$$\vdash f\ \in\ \mathsf{NMLG}\ \Gamma\ \Rightarrow\ f\ \in\ \{\ \phi\ |\ \mathsf{KG\_provable}\ \Gamma\ \phi\ \}$$

It matchs the conclusion of our induction principle exactly, so we can use `ho_match_mp_tac` to match it onto the induction principle above, and the remaining part of the proof would become straightforward.

The other half the goal, which is saying any $K\Gamma$-provable formula is in the normal modal logic $K\Gamma$:

$$\vdash \mathsf{KGproof}\ \Gamma\ p\ \Rightarrow\ \forall \phi.\ \mathsf{MEM}\ \phi\ p\ \Rightarrow\ \phi\ \in\ \mathsf{NMLG}\ \Gamma$$

is proved by induct on KGproof.

0.2.2 Chapter 2

2.1 Invariance Results

Section 2.1 of Blackburn's textbook is talking about obtaining new models from old ones without affecting modal satisfication. There are basically 3 ways to do it:

Disjoint union To define disjoint union in the HOL, noting that the disjoint union is defined on a family of models. For further convenience, we will only allow the elements of the index set to be natural numbers, so a family of models indexed by natural numbers is just a function that assigns each natural number in the index set a model. So taking the disjoint union is taking such a function and a index set as inputs, and returns a model:

$$
\begin{array}{l}
\mathsf{DU}\ (f, dom) \stackrel{\mathsf{def}}{=} \\
\quad \mathsf{<|frame}\ := \\
\quad\quad \mathsf{<|world}\ := \\
\quad\quad\quad \{\ w\ \mid\ \mathsf{FST}\ w\ \in\ dom\ \wedge\ \mathsf{SND}\ w\ \in\ (f\ (\mathsf{FST}\ w)).\mathsf{frame.world}\ \}\,; \\
\quad\quad\quad \mathsf{rel}\ := \\
\quad\quad\quad\quad (\lambda\, w_1\ w_2. \\
\quad\quad\quad\quad\quad \mathsf{FST}\ w_1\ =\ \mathsf{FST}\ w_2\ \wedge\ \mathsf{FST}\ w_1\ \in\ dom\ \wedge \\
\quad\quad\quad\quad\quad (f\ (\mathsf{FST}\ w_1)).\mathsf{frame.rel}\ (\mathsf{SND}\ w_1)\ (\mathsf{SND}\ w_2))\,\mathsf{|>}; \\
\quad \mathsf{valt}\ :=\ (\lambda\, v\ w.\ (f\ (\mathsf{FST}\ w)).\mathsf{valt}\ v\ (\mathsf{SND}\ w))\,\mathsf{|>}
\end{array}
$$

As for a model in a disjoint union, both the index of the model any world in the model are natural numbers, if w is a world of a model M_n, then the copy of the world w in the disjoint union is expressed as $n\ \otimes\ w$.So the worlds in the disjoint union are all "npair"s. And the two worlds in the disjoint union is related iff they come from the same model, and in the model that they come from they are related. The fact that two worlds n_1, n_2 in the disjoint union come from the same model is encoded as the first coordinate of them, which corresponds to the index of the model they comes from, are the same, that is, $\mathsf{nfst}\ n_1\ =\ \mathsf{nfst}\ n_2$. And we can see from the code above that the for a propostional letter v, an element n in the disjoint union is in $V(v)$ if and only if in the world indexed by $nfstn$ that n comes from, the point $nsndn$ in that model that corresponds to n is in the evalution set of v.

The special case of the union of two models M_1, M_2 is given by:

$$
\begin{array}{l}
\mathsf{M\_union}\ \mathfrak{M}_1\ \mathfrak{M}_2\ \stackrel{\mathsf{def}}{=} \\
\quad \mathsf{DU}\ ((\lambda\, n.\ \mathtt{if}\ n\ =\ 0\ \mathtt{then}\ \mathfrak{M}_1\ \mathtt{else}\ \mathfrak{M}_2), \{\ x\ \mid\ x\ =\ 0\ \vee\ x\ =\ 1\ \}\,)
\end{array}
$$

That is, taking the *dom* to be the two-element set.

We can induct on the structure of formula to prove the invarience result for disjoint unions, which says a formula is satisfies in a world in a disjoint union iff is it satisfied at the corresponding world in the model that this world comes from:

$$
\begin{array}{l}
\vdash \mathsf{FST}\ w\ \in\ dom\ \Rightarrow \\
\quad (\mathsf{satis}\ (f\ (\mathsf{FST}\ w))\ (\mathsf{SND}\ w)\ phi\ \iff \\
\quad\quad \mathsf{satis}\ (\mathsf{DU}\ (f, dom))\ w\ phi)
\end{array}
$$

generated submodel For two models $(\mathfrak{M}_1, \mathfrak{M}_2)$, we say $\mathfrak{M}_1$ is a submodel of $\mathfrak{M}_2$ if the world set of $\mathfrak{M}_1$ is a subset of that of $\mathfrak{M}_2$, and the relation and evaluation map of $\mathfrak{M}_1$ is given by that of $\mathfrak{M}_2$ restricted on the world set of $\mathfrak{M}_1$.

$$
\begin{array}{l}
\mathsf{SUBMODEL}\ \mathfrak{M}_1\ \mathfrak{M}_2\ \stackrel{\mathsf{def}}{=} \\
\quad \mathfrak{M}_1.\mathsf{frame.world}\ \subseteq\ \mathfrak{M}_2.\mathsf{frame.world}\ \wedge \\
\quad \forall\, w_1. \\
\qquad w_1\ \in\ \mathfrak{M}_1.\mathsf{frame.world}\ \Rightarrow \\
\qquad (\forall\, v.\ \mathfrak{M}_1.\mathsf{valt}\ v\ w_1\ \iff\ \mathfrak{M}_2.\mathsf{valt}\ v\ w_1)\ \wedge \\
\qquad \forall\, w_2. \\
\qquad\quad w_2\ \in\ \mathfrak{M}_1.\mathsf{frame.world}\ \Rightarrow \\
\qquad\quad (\mathfrak{M}_1.\mathsf{frame.rel}\ w_1\ w_2\ \iff\ \mathfrak{M}_2.\mathsf{frame.rel}\ w_1\ w_2)
\end{array}
$$

The generated submodel is a concept defined upon submodels. Saying a submodel $\mathfrak{M}_1$ is a generating submodel of $\mathfrak{M}_2$, we mean that for any point v in $\mathfrak{M}_1$, if the relation defined on $\mathfrak{M}_2$ relates it to a point w of $\mathfrak{M}_2$, then the w must be in the world set of $\mathfrak{M}_1$ as well:

$$
\begin{array}{l}
\mathsf{GENSUBMODEL}\ \mathfrak{M}_1\ \mathfrak{M}_2\ \stackrel{\mathsf{def}}{=} \\
\quad \mathsf{SUBMODEL}\ \mathfrak{M}_1\ \mathfrak{M}_2\ \wedge \\
\quad \forall\, w_1. \\
\qquad w_1\ \in\ \mathfrak{M}_1.\mathsf{frame.world}\ \Rightarrow \\
\qquad \forall\, w_2. \\
\qquad\quad w_2\ \in\ \mathfrak{M}_2.\mathsf{frame.world}\ \wedge\ \mathfrak{M}_2.\mathsf{frame.rel}\ w_1\ w_2\ \Rightarrow \\
\qquad\quad w_2\ \in\ \mathfrak{M}_1.\mathsf{frame.world}
\end{array}
$$

The invarience theorem of generated submodel stated in the HOL is:

$$
\begin{array}{l}
\vdash\ \mathsf{GENSUBMODEL}\ \mathfrak{M}_1\ \mathfrak{M}_2\ \wedge\ n\ \in\ \mathfrak{M}_1.\mathsf{frame.world}\ \Rightarrow \\
\quad (\mathsf{satis}\ \mathfrak{M}_1\ n\ phi\ \iff\ \mathsf{satis}\ \mathfrak{M}_2\ n\ phi)
\end{array}
$$

It says that if M_1 is a generated submodel of M_2, then for any formula ϕ, ϕ is satisfied at the world n in M_1 if and only if ϕ is satisfied at n in M_2. As from the definition of generated submodel, the world set of a generated submodel M_1 of M_2 is a subset of the world set of M_2, the corresponding point of n in M_2 is just n itself.

morphisms for modalities In this section we firstly give a bunch of definitions of morphisms between models, they are all maps between elements of world sets satisfying some extra conditions:

If $f : W \to W'$ is a function from the world set of a model $\mathfrak{M}$ to the world set of a model $\mathfrak{M}'$ then it is a homomorphism if

(i) For each propositional letter p and each element w from $\mathfrak{M}$, if $w \in V(p)$, then $f(w) \in V'(p)$.
(ii) If Rwu then $R'f(w)f(u)$.

$$\begin{array}{l}\mathsf{hom}\ f\ \mathfrak{M}_1\ \mathfrak{M}_2\ \stackrel{\mathsf{def}}{=} \\ \quad \forall\, w. \\ \qquad w\ \in\ \mathfrak{M}_1.\mathsf{frame.world}\ \Rightarrow \\ \qquad f\ w\ \in\ \mathfrak{M}_2.\mathsf{frame.world}\ \wedge \\ \qquad (\forall\, p.\ w\ \in\ \mathfrak{M}_1.\mathsf{valt}\ p\ \Rightarrow\ f\ w\ \in\ \mathfrak{M}_2.\mathsf{valt}\ p)\ \wedge \\ \qquad \forall\, u. \\ \qquad\quad u\ \in\ \mathfrak{M}_1.\mathsf{frame.world}\ \Rightarrow \\ \qquad\quad \mathfrak{M}_1.\mathsf{frame.rel}\ w\ u\ \Rightarrow \\ \qquad\quad \mathfrak{M}_2.\mathsf{frame.rel}\ (f\ w)\ (f\ u)\end{array}$$

It is a strong homomorphism if

(i) For each propositional letter p and each element w from $\mathfrak{M}$, $w \in V(p)$ iff $f(w) \in V'(p)$.
(ii) If Rwu then $R'f(w)f(u)$.

$$\begin{array}{l}\mathsf{strong\_hom}\ f\ \mathfrak{M}_1\ \mathfrak{M}_2\ \stackrel{\mathsf{def}}{=} \\ \quad \forall\, w. \\ \qquad w\ \in\ \mathfrak{M}_1.\mathsf{frame.world}\ \Rightarrow \\ \qquad f\ w\ \in\ \mathfrak{M}_2.\mathsf{frame.world}\ \wedge \\ \qquad (\forall\, p.\ w\ \in\ \mathfrak{M}_1.\mathsf{valt}\ p\ \iff\ f\ w\ \in\ \mathfrak{M}_2.\mathsf{valt}\ p)\ \wedge \\ \qquad \forall\, u. \\ \qquad\quad u\ \in\ \mathfrak{M}_1.\mathsf{frame.world}\ \Rightarrow \\ \qquad\quad (\mathfrak{M}_1.\mathsf{frame.rel}\ w\ u\ \iff\ \mathfrak{M}_2.\mathsf{frame.rel}\ (f\ w)\ (f\ u))\end{array}$$

A strong homomorphism is called an embedding if it is injective.

$$\begin{array}{l}\mathsf{embedding}\ f\ \mathfrak{M}_1\ \mathfrak{M}_2\ \stackrel{\mathsf{def}}{=} \\ \quad \mathsf{strong\_hom}\ f\ \mathfrak{M}_1\ \mathfrak{M}_2\ \wedge\ \mathsf{INJ}\ f\ \mathfrak{M}_1.\mathsf{frame.world}\ \mathfrak{M}_2.\mathsf{frame.world}\end{array}$$

And a strong morphism which is bijective is called an isomorphism.

$$\begin{array}{l}\mathsf{iso}\ f\ \mathfrak{M}_1\ \mathfrak{M}_2\ \stackrel{\mathsf{def}}{=} \\ \quad \mathsf{strong\_hom}\ f\ \mathfrak{M}_1\ \mathfrak{M}_2\ \wedge\ \mathsf{BIJ}\ f\ \mathfrak{M}_1.\mathsf{frame.world}\ \mathfrak{M}_2.\mathsf{frame.world}\end{array}$$

For descibing the concept "equivalence of modalities", we introduce a bunch of definitions about τ-theory. The τ-theory of a point in a model is the set of formulas satisfied at this point. And the τ-theory of a model is the set of the the formula that is satisfied at any point in the model.

$$\mathsf{tau\_theory}\ \mathfrak{M}\ w\ \stackrel{\mathsf{def}}{=}\ \{\ \phi\ |\ \mathsf{satis}\ \mathfrak{M}\ w\ \phi\ \}$$

$$\mathsf{tau\_theory\_model}\ \mathfrak{M}\ \stackrel{\mathsf{def}}{=}\ \{\ \phi\ |\ \forall\, w.\ w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \Rightarrow\ \mathsf{satis}\ \mathfrak{M}\ w\ \phi\ \}$$

The above definitions enables us to define modal equivalence on two levels: for points or for models: Two points from two models are called modally equivalent if their τ-theory of points are the same.

$$\mathsf{modal\_eq}\ \mathfrak{M}\ \mathfrak{M}'\ w\ w' \stackrel{\mathsf{def}}{=} \mathsf{tau\_theory}\ \mathfrak{M}\ w\ =\ \mathsf{tau\_theory}\ \mathfrak{M}'\ w'$$

Similarly, two models are called modally equivalent if their τ-theory of models are the same.

$$\mathsf{modal\_eq\_model}\ \mathfrak{M}\ \mathfrak{M}' \stackrel{\mathsf{def}}{=} \mathsf{tau\_theory\_model}\ \mathfrak{M}\ =\ \mathsf{tau\_theory\_model}\ \mathfrak{M}'$$

Here comes our invarience theorem for strong homomorphisms

$$\begin{array}{l}\vdash \mathsf{strong\_hom}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \wedge\ f\ w\ =\ w'\ \wedge\ w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge \\ \quad \mathsf{SURJ}\ f\ \mathfrak{M}.\mathsf{frame.world}\ \mathfrak{M}'.\mathsf{frame.world}\ \Rightarrow \\ \quad \mathsf{modal\_eq}\ \mathfrak{M}\ \mathfrak{M}'\ w\ w'\end{array}$$

which says if we have a surjective from a model $\mathfrak{M}$ to a model $\mathfrak{M}'$, then if the point w in $\mathfrak{M}$ is sent to w' under f, then w and w' are modally equivalent, means that they have the same τ-theory.

And the invarience theorem for the isomorphisms

$$\vdash \mathsf{iso}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \Rightarrow\ \mathsf{modal\_eq\_model}\ \mathfrak{M}\ \mathfrak{M}'$$

which says if two models are isomorphic, then they are modally equivalent. It is directly by the invarience theorem for strong homomorphisms.

We prove the first one by firstly proving a equivalent statement without mentioning τ-theory:

$$\begin{array}{l}\vdash \mathsf{strong\_hom}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \wedge\ f\ w\ =\ w'\ \wedge\ w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge \\ \quad \mathsf{SURJ}\ f\ \mathfrak{M}.\mathsf{frame.world}\ \mathfrak{M}'.\mathsf{frame.world}\ \Rightarrow \\ \quad (\mathsf{satis}\ \mathfrak{M}\ w\ \phi\ \iff\ \mathsf{satis}\ \mathfrak{M}'\ w'\ \phi)\end{array}$$

Then we quote this lemma and expand the definition of modal equivalence and τ-theory to finish the proof.

We also have the notion of bounded morphism. It is just a homomorphism plus a backward condition, (use the notations as the definition of all other morphisms)which is "If $R'f(w)v'$, then there exists a v in $\mathfrak{M}$ such that Rwv and $f(v) = v'$".

$$
\begin{array}{l}
\mathsf{bounded\_mor}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \stackrel{\mathsf{def}}{=} \\
\quad \forall\, w. \\
\qquad w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \Rightarrow \\
\qquad f\ w\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge \\
\qquad (\forall\, a.\ \mathsf{satis}\ \mathfrak{M}\ w\ (\mathsf{VAR}\ a)\ \Longleftrightarrow\ \mathsf{satis}\ \mathfrak{M}'\ (f\ w)\ (\mathsf{VAR}\ a))\ \wedge \\
\qquad (\forall\, v. \\
\qquad\quad v\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ \mathfrak{M}.\mathsf{frame.rel}\ w\ v\ \Rightarrow \\
\qquad\quad \mathfrak{M}'.\mathsf{frame.rel}\ (f\ w)\ (f\ v))\ \wedge \\
\qquad \forall\, v'. \\
\qquad\quad v'\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge\ \mathfrak{M}'.\mathsf{frame.rel}\ (f\ w)\ v'\ \Rightarrow \\
\qquad\quad \exists\, v.\ v\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ \mathfrak{M}.\mathsf{frame.rel}\ w\ v\ \wedge\ f\ v\ =\ v'
\end{array}
$$

If we have a surjective bounded morphism from M to M', then M' is called the bounded morphic image of M under f. That is:

$$
\begin{array}{l}
\mathsf{bounded\_mor\_image}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \stackrel{\mathsf{def}}{=} \\
\quad \mathsf{bounded\_mor}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \wedge\ \mathsf{SURJ}\ f\ \mathfrak{M}.\mathsf{frame.world}\ \mathfrak{M}'.\mathsf{frame.world}
\end{array}
$$

The invarience theorem for bounded morphism is:

$$
\begin{array}{l}
\vdash\ \mathsf{bounded\_mor}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \wedge\ w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \Rightarrow \\
\quad (\mathsf{satis}\ \mathfrak{M}\ w\ \phi\ \Longleftrightarrow\ \mathsf{satis}\ \mathfrak{M}'\ (f\ w)\ \phi)
\end{array}
$$

It states that if f is a bounded morphism from $\mathfrak{M}$ to $\mathfrak{M}'$, then for any world w in $\mathfrak{M}$, w and $f(w)$ are modally equivalent. Same as the invarience theorem for the previous constructions, it is proved by structual induction on formulas.

Now we give a definition of a important class of models, the tree-like models. Here we define a tree as a type of frame, so a tree-like model is just a model defined upon a frame which is a tree.

We say a frame S is a tree with the root r if the r is a point in the frame, satisfying:

(1) For any world in S, the root r is linked to it via the transitive closure of the relation defined on S.
(2) There is no point in S that is linked to the root via the relation on S.
(3) For any non-root point in S, there exists a unique point that is linked to it via the relation on S.

$$
\begin{array}{l}
\mathsf{tree}\ S\ r\ \stackrel{\mathsf{def}}{=} \\
\quad r\ \in\ S.\mathsf{world}\ \wedge \\
\quad (\forall\, t.\ t\ \in\ S.\mathsf{world}\ \Rightarrow\ (\mathsf{RESTRICT}\ S.\mathsf{rel}\ S.\mathsf{world})^*\ r\ t)\ \wedge \\
\quad (\forall\, r_0.\ r_0\ \in\ S.\mathsf{world}\ \Rightarrow\ \neg S.\mathsf{rel}\ r_0\ r)\ \wedge \\
\quad \forall\, t.\ t\ \in\ S.\mathsf{world}\ \wedge\ t\ \neq\ r\ \Rightarrow\ \exists!\, t_0.\ t_0\ \in\ S.\mathsf{world}\ \wedge\ S.\mathsf{rel}\ t_0\ t
\end{array}
$$

This definition is slightly different from the definition of the tree in the textbook, which does not say $(\forall r_0 \in S.world)(\neg S.rel\ r_0 r)$, but states that a

tree has no loop as $(\forall t)\neg S^+\ t\ t$. To unify these two versions of definitions, we need to prove a tree has no loop as the following lemma:

$$\vdash \mathsf{tree}\ s\ r\ \Rightarrow\ \forall\, t_0\ t.\ (\mathsf{RESTRICT}\ s.\mathsf{rel}\ s.\mathsf{world})^+\ t_0\ t\ \Rightarrow\ t_0\ \neq\ t$$

This is proved by using `ho_match_mp_tac` with the induction principle `RTC_STRONG_INDUCT_RIGHT1`:

$$\begin{array}{l}\vdash (\forall\, x.\ P\ x\ x)\ \wedge\ (\forall\, x\ y\ z.\ P\ x\ y\ \wedge\ R^*\ x\ y\ \wedge\ R\ y\ z\ \Rightarrow\ P\ x\ z)\ \Rightarrow\\ \quad \forall\, x\ y.\ R^*\ x\ y\ \Rightarrow\ P\ x\ y\end{array}$$

Quite related, we have a definition of rooted model. A rooted model is a submodel of a model generated by a single point. For a rooted model, it would definitly be a submodel for some model, and the world set of a rooted model is exactly all the worlds that a world links to via the transitive closure of the relation of the model. $\mathsf{rooted\_model}\ \mathfrak{M}\ x\ \mathfrak{M}'$ reads "$\mathfrak{M}$ is a rooted model with root x, and it is a submodel of $\mathfrak{M}'$".

$$\begin{array}{l}\mathsf{rooted\_model}\ \mathfrak{M}\ x\ \mathfrak{M}'\ \stackrel{\mathsf{def}}{=}\\ \quad x\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge\\ \quad (\forall\, a.\\ \qquad a\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \Longleftrightarrow\\ \qquad\quad a\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge\\ \qquad\qquad (\mathsf{RESTRICT}\ \mathfrak{M}'.\mathsf{frame.rel}\ \mathfrak{M}'.\mathsf{frame.world})^*\ x\ a)\ \wedge\\ \quad (\forall\, n_1\ n_2.\\ \qquad n_1\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ n_2\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \Rightarrow\\ \qquad (\mathfrak{M}.\mathsf{frame.rel}\ n_1\ n_2\ \Longleftrightarrow\\ \qquad\quad \mathsf{RESTRICT}\ \mathfrak{M}'.\mathsf{frame.rel}\ \mathfrak{M}'.\mathsf{frame.world}\ n_1\ n_2))\ \wedge\\ \quad \forall\, v\ n.\ \mathfrak{M}.\mathsf{valt}\ v\ n\ \Longleftrightarrow\ \mathfrak{M}'.\mathsf{valt}\ v\ n\end{array}$$

There is a significant result pointing out the importance of tree-like models and the connection between tree-like models, rooted models and bounded morphisms:

It says for any rooted model $\mathfrak{M}$, there is a tree-like model T such that $\mathfrak{M}$ is a bounded morphic image of T under some bounded morphism. So together with the invarience result for bounded morphisms, any satisfiable τ-formula is satisfiable in a tree-like model.

So to prove it, we need to specify such a bounded morphism and a tree-like model. For a rooted model $\mathfrak{M}$ whose root it x, the corresponding tree-like model is:

```
bounded_preimage_rooted 𝔐 x ≝
 <|frame :=
  <|world :=
     { l |
      HD l = x ∧ LENGTH l > 0 ∧
      ∀ m.
         m < LENGTH l − 1 ⇒
         RESTRICT 𝔐.frame.rel 𝔐.frame.world (EL m l)
          (EL (m + 1) l) };
    rel :=
     (λ l1 l2.
         LENGTH l1 + 1 = LENGTH l2 ∧
         RESTRICT 𝔐.frame.rel 𝔐.frame.world (LAST l1)
          (LAST l2) ∧
         ∀ m. m < LENGTH l1 ⇒ EL m l1 = EL m l2)|>;
  valt := (λ v n. 𝔐.valt v (LAST n))|>
```

That is, the worlds in this model are lists, encoded as natural numbers using `numlistTheory`. The definition actually says a list $(u_0, u_1, ..., u_n)$ is in the world of this model iff the list is nonempty, begining with the root x, and recording a path in M from the root x to u_n. A list l_1 is related to a list $l2$ iff $l2$ is one-element longer then l_1, if we put l_1 and l_2 in parallel and compare their members, they are all the same until the l_1 is end, and the last element of l_1 is linked to the last element of l_2 via the relation in $\mathfrak{M}$. And the evaluation map is defined by a list $l_1 \in V_{rooted}(p)$ iff the last element of $l_1 \in V(p)$ where V is the evaluation map of $\mathfrak{M}$. The model $\mathfrak{M}$ is then the bounded morphic image of it under the bounded morphism defined by taking the last element of the list, this is the function "nlast".

So we have two things to prove. One thing is that the model defined above is indeed a tree:

Beyond the works about numlistTheory(Here I have omitted all the works about the `numlistTheory`, otherwise the description of this part would be even longer then the already existing article.), The trickest part is to prove the unique existence of the predecessor of a non-root element. For a non-root element, its predecessor is the list obtained by discarding the last element of the list. discarding the last element of a numlist is defined as a function nfront, and the following lemmas proves that for a list t, nfront t is indeed a predecessor of t by proving that it is both in the world of our defined model, and it is linked to t by the relation on the model.

For proving that the predecessor is unique, note that an numlist is uniquely defined by the members in it. so once we conclude that if a list is a predecessor of a given list t, then each member of it is fixed, then we are done. And the information that "given a non-root element, each member of its predecessor is fixed" can be extracted from the third clauses of the definition of relation in `bounded_preimage_rooted_def`.

Having proved the first half. The other thing is that for the rooted model M with root x it is indeed the bounded morphic image of the model we construct above under the bounded morphism "nlast".

For the proof of this, rewriting with the definition of bounded morphic image gives four subgoals. Among them, a non-trivial one is the fact that if n is a list in the world set of the model we have defined, then $\mathsf{nlast}\ n$ is in the world set of M. To deal with it, note that $\mathsf{nlast}\ n\ =\ \mathsf{nel}\ (\mathsf{LENGTH}\ (\mathsf{listOfN}\ n)\ -\ 1)\ n$, then we can prove a lemma that for a numlist in our constructed model, every member of the list is in the world of $\mathfrak{M}$:

to proves the desired fact.

Another non-trivial part is the surjectiveness of the map nlast, this is proved as a lemma:

by induction on the reflexive transitive closure relation. To actually apply the above lemma into our proof. We also need to prove

which tells us for any element in a rooted model, it is linked to the root by the reflextive transitive closure. So we can get the anticedent of our lemma holds, which allows us to apply it.

With the two above theorems proved, the main theorem can be solved easily by applying `qexists_tac` and `metis_tac` with the above two lemmas.

2.2 Bisimulations

This section talks about bisimulations, which is defined as a relation between elements in the world sets of two models satisfying some conditions:

$$
\begin{array}{l}
\mathsf{bisim}\ Z\ \mathfrak{M}\ \mathfrak{M}'\ \stackrel{\mathsf{def}}{=} \\
\quad \forall\, w\ w'. \\
\qquad w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ w'\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge\ Z\ w\ w'\ \Rightarrow \\
\qquad (\forall\, a.\ \mathsf{satis}\ \mathfrak{M}\ w\ (\mathsf{VAR}\ a)\ \Longleftrightarrow\ \mathsf{satis}\ \mathfrak{M}'\ w'\ (\mathsf{VAR}\ a))\ \wedge \\
\qquad (\forall\, v. \\
\qquad\quad v\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ \mathfrak{M}.\mathsf{frame.rel}\ w\ v\ \Rightarrow \\
\qquad\quad \exists\, v'.\ v'\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge\ Z\ v\ v'\ \wedge\ \mathfrak{M}'.\mathsf{frame.rel}\ w'\ v')\ \wedge \\
\qquad \forall\, v'. \\
\qquad\quad v'\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge\ \mathfrak{M}'.\mathsf{frame.rel}\ w'\ v'\ \Rightarrow \\
\qquad\quad \exists\, v.\ v\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ Z\ v\ v'\ \wedge\ \mathfrak{M}.\mathsf{frame.rel}\ w\ v
\end{array}
$$

As we can read from the code, for a relation Z between models M and M' to be a bisimulation, the worlds which are related must satisfies the same propositional letters, and the forward consition:

If wZw' and Rwv, then there exists a world v' in M' such that vZv' and $R'w'v'$.

And the backward condition:

If wZw' and $R'w'v'$, then there exists a world v in M such that vZv' and Rwv.

If two points in two models are related by a bisimulation, then we call the two points are bisimilar:

We can also say that two models are bisimilar if there is a bisimulation between them:

In fact, there are many situations where we can obtain a bisimulation:

(1) If $\mathfrak{M} \cong \mathfrak{M}'$, then $\mathfrak{M}$ is bisimilar to $\mathfrak{M}'$. (The bisimulation is given by relating a world to its image under the isomorphism.)

$$\vdash \mathsf{iso}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \Rightarrow\ \mathsf{bisim\_model}\ \mathfrak{M}\ \mathfrak{M}'$$

(2) For a disjoint union $\mathsf{DU}\ (f, dom)$ of models, any model $f\ i$ where i is in the dom is bisimilar to the unioned model. (The bisimulation is given by relating a world to its copy in the disjoint union.)

$$\begin{array}{l}\vdash i\ \in\ dom\ \wedge\ w\ \in\ (f\ i).\mathsf{frame.world}\ \Rightarrow \\ \quad \mathsf{bisim\_world}\ (f\ i)\ (\mathsf{DU}\ (f, dom))\ w\ (i, w)\end{array}$$

(3) If $\mathfrak{M}$ is a generated submodel of $\mathfrak{M}'$, then $\mathfrak{M}$ is bisimilar to $\mathfrak{M}'$. (The bisimulation is given by relating a world to itself.)

$$\begin{array}{l}\vdash \mathsf{GENSUBMODEL}\ \mathfrak{M}\ \mathfrak{M}'\ \Rightarrow \\ \quad \forall\, w.\ w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \Rightarrow\ \mathsf{bisim\_world}\ \mathfrak{M}\ \mathfrak{M}'\ w\ w\end{array}$$

(4) If $\mathfrak{M}'$ is a bounded morphic image of $\mathfrak{M}$ under the bounded morphism f, then $\mathfrak{M}$ is bisimilar to $\mathfrak{M}'$. (The bisimulation is given by relating to a world to its image under the bounded morphism.)

$$\begin{array}{l}\vdash \mathsf{bounded\_mor\_image}\ f\ \mathfrak{M}\ \mathfrak{M}'\ \Rightarrow \\ \quad \forall\, w.\ w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \Rightarrow\ \mathsf{bisim\_world}\ \mathfrak{M}\ \mathfrak{M}'\ w\ (f\ w)\end{array}$$

Similar to the last section, there is also an invariance theorem for bisimulations. It says that if $\mathfrak{M}$ and $\mathfrak{M}'$ are models, then if two worlds w, w' in the two models are linked by a bisimulation, then they are modally equivalent.

$$\vdash \mathsf{bisim\_world}\ \mathfrak{M}\ \mathfrak{M}'\ w\ w'\ \Rightarrow\ \mathsf{modal\_eq}\ \mathfrak{M}\ \mathfrak{M}'\ w\ w'$$

A natural question is that whether the converse of the above theorem holds. That is, if two worlds in two models are modally equivalent, whether they are bisimilar or not. In general, it does not. But the converse of the above theorem does hold for a special kind of models, called image-finite models. This kind of model is defined as the model has a special property that each point in the model is only related to a finite number of worlds, as defined below:

$$\begin{array}{l}\mathsf{image\_finite}\ \mathfrak{M}\ \stackrel{\mathsf{def}}{=} \\ \ \forall\, x. \\ \quad x\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \Rightarrow \\ \quad \mathsf{FINITE}\ \{\ y\ |\ y\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ \mathfrak{M}.\mathsf{frame.rel}\ x\ y\ \}\end{array}$$

The invarience theorem of this kind of model is called Hennessy-Milner Theorem:

It says that for worlds in image-finite models, modal equivalence is equivalent to being linked by a bisimulation. Half of this theorem is just a special case of the invarience theorem of bisimulations, the part that requires a proof is:

$$\begin{array}{l}\vdash \mathsf{image\_finite}\ \mathfrak{M}\ \wedge\ \mathsf{image\_finite}\ \mathfrak{M}'\ \wedge\ w\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge \\ \quad w'\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \Rightarrow \\ \quad (\forall\,\phi.\ \mathsf{satis}\ \mathfrak{M}\ w\ \phi\ \iff\ \mathsf{satis}\ \mathfrak{M}'\ w'\ \phi)\ \Rightarrow \\ \quad \mathsf{bisim\_world}\ \mathfrak{M}\ \mathfrak{M}'\ w\ w'\end{array}$$

We will prove that for two image-finite models, if two worlds are modally equivalent, then the relation Z that wZw' iff w and w' are modally equivalent is a bisimulation.

If we use `qexists_tac` to specify the bisimulation $\lambda\, n_1\ n_2.\ \forall\,\phi.\ \mathsf{satis}\ \mathfrak{M}\ n_1\ \phi\ \iff\ \mathsf{satis}\ \mathfrak{M}'\ n_2\ \phi$ and expand the definition of bisimulation of proving it, the HOL would ask us to prove two subgoals: both the forward condition and the backward consition. But the bisimulation we use here is actually special: it is a equivalence relation which is symmetric, so the forward and backward condition makes no difference, means that it suffices to prove only one of them. To convince the HOL that we just want prove only one of them, we prove this lemma:

The $P\ \mathfrak{M}\ N$ here is the $\mathsf{image\_finite}\ \mathfrak{M}\ \wedge\ \mathsf{image\_finite}\ \mathfrak{M}'$ in the theorem, and $Z\ \mathfrak{M}\ N\ u\ v$ corresponds to $\forall\,\phi.\ \mathsf{satis}\ \mathfrak{M}'\ n_2\ \phi\ \iff\ \mathsf{satis}\ \mathfrak{M}\ n_1\ \phi$. So once we prove this theorem, we can match it to the original statement with `ho_match_mp_tac` to give:

$$\begin{array}{l}(\forall\,\mathfrak{M}'\ \mathfrak{M}. \\ \quad \mathsf{image\_finite}\ \mathfrak{M}'\ \wedge\ \mathsf{image\_finite}\ \mathfrak{M}\ \Rightarrow \\ \quad \mathsf{image\_finite}\ \mathfrak{M}\ \wedge\ \mathsf{image\_finite}\ \mathfrak{M}')\ \wedge \\ (\forall\,\mathfrak{M}'\ \mathfrak{M}\ n_2\ n_1. \\ \quad (\forall\,\phi.\ \mathsf{satis}\ \mathfrak{M}'\ n_2\ \phi\ \iff\ \mathsf{satis}\ \mathfrak{M}\ n_1\ \phi)\ \Rightarrow \\ \quad \forall\,\phi.\ \mathsf{satis}\ \mathfrak{M}\ n_1\ \phi\ \iff\ \mathsf{satis}\ \mathfrak{M}'\ n_2\ \phi)\ \wedge \\ \forall\,\mathfrak{M}\ \mathfrak{M}'. \\ \quad \mathsf{image\_finite}\ \mathfrak{M}\ \wedge\ \mathsf{image\_finite}\ \mathfrak{M}'\ \Rightarrow \\ \quad \forall\,n_1\ n_2. \\ \qquad n_1\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ n_2\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge \\ \qquad (\forall\,\phi.\ \mathsf{satis}\ \mathfrak{M}\ n_1\ \phi\ \iff\ \mathsf{satis}\ \mathfrak{M}'\ n_2\ \phi)\ \Rightarrow \\ \qquad (\forall\,a.\ \mathsf{satis}\ \mathfrak{M}\ n_1\ (\mathsf{VAR}\ a)\ \iff\ \mathsf{satis}\ \mathfrak{M}'\ n_2\ (\mathsf{VAR}\ a))\ \wedge \\ \qquad \forall\,n_1'. \\ \qquad\quad n_1'\ \in\ \mathfrak{M}.\mathsf{frame.world}\ \wedge\ \mathfrak{M}.\mathsf{frame.rel}\ n_1\ n_1'\ \Rightarrow \\ \qquad\quad \exists\,n_2'. \\ \qquad\qquad n_2'\ \in\ \mathfrak{M}'.\mathsf{frame.world}\ \wedge \\ \qquad\qquad (\forall\,\phi.\ \mathsf{satis}\ \mathfrak{M}\ n_1'\ \phi\ \iff\ \mathsf{satis}\ \mathfrak{M}'\ n_2'\ \phi)\ \wedge \\ \qquad\qquad \mathfrak{M}'.\mathsf{frame.rel}\ n_2\ n_2'\end{array}$$

After `rpt strip_tac`, we can see many of the subgoals are trivial, and the only remaining task if prove one of the forward/backward condtion. So we can save labour on repeating the same proof.

The thing remains to prove to is that if n_1 in $\mathfrak{M}$,n_2 in $\mathfrak{M}'$ are modally equivalent, then for all n_1' in $\mathfrak{M}$ with Rn_1n_1', there exists an $n_2' \in M'$ such that $R'n_2n_2'$ and n_1' and n_2' are modally equivalent. It is proved by assuming that there exists a point in n_1' such that Rn_1n_1', and for all n_2' in $\mathfrak{M}'$ such that $R'n_2n_2'$, n_1' is not modally equivalent to n_2', then we will reach a contradiction by proving then n_1 and n_2 would not be modally equivalent. To conclude this, we need to find out a formula that is satisfied at n_1 but not n_2. It turns out that the formula we need is one with a diamond in the front. So it suffices to prove that there is a formula that is satisfied at n_1', but is not satisfied at any the world that is related to n_2. It turns out that we can find a formula which is a big conjunction that satisfies this condition. But the problem is: We have not got a definition of big conjunction by hand. To tackle this problem, we choose to do not use `qexists_tac` to specify an explict formula, but to prove its existence as a lemma.

We are given that $\mathfrak{M}$,$\mathfrak{M}'$ are both image finite, so the set of the worlds related to n_2 in $\mathfrak{M}'$ is finite. Also the set of such worlds is non-empty, since otherwise the formula $\Box\bot$ is vacuously true at n_2 but is false at n_1.

So once this lemma is proved, we can apply it to our case where s is the set of worlds that are related to n_2, v is the n_1', and then the "psi" in the conclusion gives the desire formula which gives the contradiction. Then by the argument above, we have proved the main theorem.