

Monadic Second-Order Logic in HOL: Deep and Shallow with Automated Faithfulness (Extended Preprint)

Christoph Benzmueller^{1,2}[0000-0002-3392-3093][★] and Daniel Kirchner¹[0000-0001-9229-1148]

¹ University of Bamberg, Bamberg, Germany

christoph.benzmueller@uni-bamberg.de, daniel.kirchner@uni-bamberg.de

² Freie Universität Berlin, Berlin, Germany

Abstract. In Isabelle/HOL, we apply the deep-and-shallow embedding methodology of our prior work to *monadic second-order logic* (MSO). Three embeddings are developed side by side: a *deep* embedding (an inductive datatype with an explicit satisfaction relation); a *maximal-shallow* embedding that translates the connectives and quantifiers directly into HOL, carrying the interpretation and both assignments as explicit arguments; and a *minimal-shallow* embedding — a locale that fixes those parameters, collapsing the formula type to *bool*. The enabling new ingredient is a *two-sorted* substitution apparatus — capture-avoiding substitution, renaming, and a substitution lemma per namespace — in which each binder is transparent for the other; faithfulness of all three embeddings is mechanised and automated. Our central contribution is a fully mechanised two-sorted downward Löwenheim–Skolem theorem: the minimal embedding recovers deep validity relative to the (countable) assignment ranges, and this range-relative reading is shown to coincide with the general (Henkin-style) reading of MSO, whereas the standard reading is provably stronger, witnessed by comprehension. Both readings are nonetheless recovered from the minimal embedding, differing only in the admitted interpretations: all of them for the general reading, only the *elementary substructures* of the full model for the standard. We further exercise the embeddings on classical MSO landmarks: the Boolean-closure and graph schemata hold under the full second-order domain yet fail in the minimal embedding, making the dichotomy concrete, while reachability and 2-colorability are refuted throughout.

Keywords: Monadic second-order logic · Deep and shallow embedding · Isabelle/HOL · Henkin semantics · Automated faithfulness

1 Motivation and Introduction

Monadic second-order logic (MSO) sits at the heart of a body of results connecting logic, automata, and graph/tree structure theory [14,22,16]. It extends

[★] Dedicated to David Basin, whom I first met in Saarbrücken in the mid-1990s, over a shared enthusiasm for automated reasoning, formal methods, and expressive logics.

first-order relational logic with a single, but powerful, additional resource: quantification over *monadic* predicates, i.e. over sets of individuals.

On the algorithmic side, a substantial line of work led by David Basin approaches MSO through automata and decision procedures: Basin and Klarlund use the weak monadic second-order theory of one successor (WS1S) and canonical string automata to specify and verify hardware, obtaining a decision procedure with counter-model generation [4,3], and Ayari and Basin develop bounded model construction for monadic second-order logics [1]. That line targets decidable fragments over strings and trees; the present work is complementary — it embeds full and general (Henkin-style) MSO into classical higher-order logic for interactive proof and metatheory, rather than deciding a fixed fragment.

Embeddings in classical higher-order logic (HOL) [15] are a versatile host for non-classical and expressive logics: treating HOL as a universal meta-logic, a wide range of object logics — modal, deontic, epistemic, and others — inherit its semantic infrastructure and proof automation. This is the basis of *LogiKey* [8,9], a logic-pluralistic knowledge representation methodology in HOL, from computational metaphysics to ethical and legal reasoning; it grew out of the early modal-logic embeddings developed together with Paulson [10,11].

Shallow embeddings, which represent object-level formulae directly as HOL terms (for MSO, by the standard translation of connectives and quantifiers), give immediate access to Isabelle/HOL’s proof automation — the prover **sledgehammer**, the finite-countermodel finder **nitpick**, and **auto**. *Deep* embeddings, in which formulae are elements of an inductively defined datatype, are needed for metalogical reasoning about syntax, substitution, or proof systems.

Shallow semantical embeddings of propositional and quantified modal logics in HOL were developed by Benzmüller and Paulson [10,11], initially without a deep embedding or a mechanised faithfulness proof. We have recently shown [12] that *both* embeddings can be carried out simultaneously, in a single HOL theory, with mutual faithfulness proofs automated. That development, however, only treats propositional modal logic, whose binders are modal operators carrying no variable binding. The same methodology now spans a range of logics; we develop it here for monadic second-order logic (MSO). We take up MSO not for expressive strength but for the phenomena of its second-order quantifier — above all the standard-versus-general (Henkin) distinction behind our central result — and, on the technical side, for its two-sorted binding: quantifying over individuals and sets of individuals at once, MSO’s first- and second-order existentials are the two binders our apparatus must handle.

Contributions. We contribute **MSOinHOL**, an Isabelle/HOL development of MSO with binary relational atoms, monadic predication atoms, the propositional connectives, a first-order existential quantifier, and a monadic second-order existential quantifier. The particular contributions include:

1. A *deep* embedding (datatype $\mathcal{F}$) and two *shallow* ones: a maximal embedding carrying explicit dependencies on the interpretation, the two domains, and the two assignments; and a minimal embedding that fixes the interpretation and assignments as locale parameters, collapsing the formula type to *bool*.

2. The minimal embedding is an Isabelle/HOL *locale* `MinS`; it admits a faithfulness theorem `FaithfulMS_all` stating that quantifying over all locale interpretations recovers exactly deep validity relative to the assignment ranges. Faithfulness of the maximal-shallow embedding with respect to the deep one (`FaithfulSD`) is likewise established and automated.
3. A *two-sorted* substitution apparatus for the deep embedding — free/bound/fresh predicates, capture-avoiding substitution, alphabetic renaming, the substitution lemma, and size-based induction — developed once per variable namespace, with each binder *transparent* for the other.
4. Experiments confirming tautologies, the monadic *comprehension* schema (proved for an arbitrary formula, generalising two ad-hoc instances), and `nitpick` countermodels for invalid schemata.
5. A suite of *classical MSO landmarks* — Boolean closure [14,22], monadic graph operations [16], reachability [4], and 2-colorability [22] — replicated across all three embeddings; the closure and graph schemata hold under the full second-order domain but *fail* in the general minimal embedding (and are recovered by its elementary refinement), while reachability and 2-colorability are non-theorems refuted throughout, turning the standard-vs-general dichotomy into a table of validity proofs, `nitpick`-found countermodels, and entailed cases.
6. The converse: range-relative validity *coincides* with general (Henkin-style) MSO validity, by a two-sorted Löwenheim–Skolem construction, while the standard reading is provably stronger — all formally proved.

Organisation. §2 fixes notation for HOL and §3 recalls MSO. §4 presents the three embeddings – deep, maximal-shallow, and the locale-based minimal-shallow (§4.4). §5 outlines the two-sorted substitution machinery, §6 the faithfulness theorems, and §7 the surjectivity problem. §8 reports experiments and §9 concludes with related work. The complete formal development is published as the Archive of Formal Proofs entry `MSOinHOL` [6]. Rendered theory sources appear in App. B and the full two-sorted Löwenheim–Skolem proof in App. A.

2 Classical higher-order logic

We adopt the presentation of HOL given in [12] without modification: simply typed λ -calculus over base types o (truth values) and D (individuals), Henkin semantics, and the standard denotation $\llbracket \cdot \rrbracket$ derived from a frame $\mathcal{D}$ and an interpretation I . The validity relation $\models_{\text{HOL}}$ is sound and complete for Henkin models [18],³ and Isabelle/HOL [20] mechanises a fragment of HOL sufficient for all that follows. In particular, the second-order domain of MSO is the HOL function type $D \Rightarrow o$, so monadic sets (resp. predicates) are first-class HOL objects.

³ The landscape of standard models, Henkin models, and Andrews’ weaker *v*-complexes, and the role of extensionality in fixing the class of Henkin models, is analysed by Benz Müller, Brown, and Kohlhasse [5]; the shallow embeddings below inherit HOL’s functional and Boolean extensionality.

3 Monadic second-order logic (MSO)

The syntax of MSO, given a (nonempty) signature R of binary relation symbols, a denumerable set V of individual (first-order) variables, and a denumerable set V_2 of monadic (second-order) variables, is fixed by the grammar

$$\varphi, \psi := r(x, y) \mid X(x) \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid \exists X.\varphi \quad (1)$$

for relation symbols $r \in R$, individual variables $x, y \in V$, and monadic predicate variables $X \in V_2$; the atom $X(x)$ asserts that the monadic predicate X holds of the individual x (extensionally, that x lies in the set denoted by X). The further connectives $\vee, \supset, \leftrightarrow$, the first-order universal $\forall x$, and the second-order universal $\forall X$ are defined as usual by duality.

A *model* is a tuple $\langle I, D, E \rangle$, where D is a nonempty domain of individuals, E is a nonempty collection of *admissible* monadic predicates over D (the second-order domain; extensionally a family $E \subseteq \mathcal{P}(D)$ of subsets), and the interpretation I assigns to each relation symbol r a binary relation $I(r) \subseteq D \times D$. A *first-order assignment* $g: V \rightarrow D$ maps each individual variable to a domain element, and a *second-order assignment* $G: V_2 \rightarrow E$ maps each monadic predicate variable to an admissible predicate in E ; $g[x \leftarrow d]$ and $G[X \leftarrow S]$ denote the obvious updates.

Truth in a model is given recursively by

$$\begin{aligned} \langle I, D, E \rangle, g, G &\models r(x, y) && \text{iff } I(r)(g(x), g(y)) \\ \langle I, D, E \rangle, g, G &\models X(x) && \text{iff } G(X) \text{ holds of } g(x) \\ \langle I, D, E \rangle, g, G &\models \neg\varphi && \text{iff not } \langle I, D, E \rangle, g, G \models \varphi \\ \langle I, D, E \rangle, g, G &\models \varphi \wedge \psi && \text{iff } \langle I, D, E \rangle, g, G \models \varphi \text{ and } \langle I, D, E \rangle, g, G \models \psi \\ \langle I, D, E \rangle, g, G &\models \exists x.\varphi && \text{iff } \exists d \in D. \langle I, D, E \rangle, g[x \leftarrow d], G \models \varphi \\ \langle I, D, E \rangle, g, G &\models \exists X.\varphi && \text{iff } \exists S \in E. \langle I, D, E \rangle, g, G[X \leftarrow S] \models \varphi \end{aligned}$$

A formula φ is *valid* iff $\langle I, D, E \rangle, g, G \models \varphi$ for every model, every g into D , and every G into E . When $E = \mathcal{P}(D)$ (every monadic predicate is admissible) the second-order quantifier is the *standard* (full) one; restricting E yields the various *general* (Henkin-style) readings. The principle bridging the two is *comprehension* — that every property definable by a formula determines an admissible set; it holds by fiat when $E = \mathcal{P}(D)$ but can fail once E is restricted. This is the *monadic* fragment: only monadic predicate variables (not relation variables) may be quantified, and a unary property of individuals is rendered directly as a monadic predicate applied to its argument.

Scope. For minimality we restrict the first-order signature to *binary* relation symbols and omit primitive equality; unary properties are rendered directly as monadic predications $X(x)$. Thus the formalisation covers the equality-free binary-relational fragment of MSO, which suffices for all results below.

4 Three embeddings of MSO in HOL

We now present the three embeddings side by side, after fixing the preliminaries shared by all of them.

4.1 Preliminaries

Preliminaries shared by all embeddings (theory `MSOinHOL_preliminaries.thy`, Fig. 6) declare the type D (individuals) and the type synonyms $R := \text{nat}$ (relation symbols), $V := \text{nat}$ (first-order variables), $V_2 := \text{nat}$ (second-order variables), together with $\mathcal{R}, \mathcal{I}$, the first-order assignment type $\mathcal{E} := V \Rightarrow D$, the second-order assignment type $\mathcal{G} := V_2 \Rightarrow (D \Rightarrow \text{bool})$, and the two domain-restriction types $\mathcal{D} := D \Rightarrow \text{bool}$ and $\mathcal{P} := (D \Rightarrow \text{bool}) \Rightarrow \text{bool}$. Two variable-assignment update operators are introduced: $g[x \leftarrow d]$ for the first-order assignment and $G\langle X \leftarrow S \rangle$ for the second-order one, each with its companion lemmata (L1–L4 and M1–M4). The set-as-predicate operations $\sqsubseteq, \sqcup, \text{Range}, \text{Univ}, \text{into}, \text{onto}$ are polymorphic and hence shared between both sorts.

4.2 Deep embedding

The deep embedding (theory `MSOinHOL_deep.thy`, Fig. 1) is a single inductive datatype $\mathcal{F}$ whose constructors mirror the grammar (1): `AtmD` (relational atom $r^d(x, y)$), `PrdD` (monadic predication atom $X^d(x)$), `NegD`, `AndD`, the first-order `ExD` ($\exists^d x. \varphi$), and the second-order `ExD2` ($\exists_2^d X. \varphi$). The relative-truth predicate $\langle I, D, E \rangle, g, G \models^d \varphi$ is declared in mixfix syntax and defined by primitive recursion: the predication clause evaluates the monadic predicate $G(X)$ at the individual $g(x)$; the first-order existential ranges over the explicit domain D ; the second-order existential ranges over the explicit collection E of admissible sets; the rest is standard. Validity `valD` closes universally over models, over domain-respecting first-order assignments (`g into D`), and over domain-respecting second-order assignments (`G into E`). An auxiliary full-domain notion $\models^{d'}$ fixes $D = \text{Univ}$ and $E = \text{Univ}$ and is a convenience for `nitpick`; since $\models^d \varphi$ implies $\models^{d'} \varphi$, any counterexample for $\models^{d'}$ refutes $\models^d$ as well.

Reading the clauses. The two existential clauses are the only places where the model is consulted nontrivially: each quantifies a witness over its domain (d over D , S over E) and re-evaluates under the corresponding assignment update. All remaining connectives ($\vee, \supset, \leftrightarrow$, and the two universals) are definitional abbreviations, unfolded automatically through the rewrite bag `DefD`; the same discipline (`DefS`, `DefM`) governs the two shallow embeddings, so the three notions of validity differ only in their primitives.

4.3 Maximal shallow embedding

The maximal shallow embedding (theory `MSOinHOL_shallow.thy`, Fig. 2) lifts every primitive of MSO to a λ -term abstracted over the explicit dependencies

```

1 theory MSOinHOL_deep imports MSOinHOL_preliminaries
2 begin
3   <Deep embedding of monadic second-order logic (MSO) in HOL.>
4   <Two binders: <math>\exists^d</math> ranges over individuals; <math>\exists^d_2</math> ranges over monadic predicates.>
5   <X^d(x) is the monadic predication atom "predicate X holds of individual x".>
6   datatype  $\mathcal{F}$  = AtmD R V V ("_{-}^d(_{-})")
7     | PrdD V2 V ("_{-}^d(_{-})")
8     | NegD  $\mathcal{F}$  ("_{-}^d _" [58]59)
9     | AndD  $\mathcal{F}$   $\mathcal{F}$  (infixr " $\wedge^d$ " 56)
10    | ExD V  $\mathcal{F}$  ("_{-}^d _")
11    | ExD2 V2  $\mathcal{F}$  ("_{-}^d_2 _")
12   <Further connectives as derived definitions.>
13   definition OrD (infixr " $\vee^d$ " 54) where " $\varphi \vee^d \psi \equiv \neg^d(\neg^d \varphi \wedge^d \neg^d \psi)$ "
14   definition ImpD (infixr " $\supset^d$ " 55) where " $\varphi \supset^d \psi \equiv \neg^d \varphi \vee^d \psi$ "
15   definition IffD (infixr " $\longleftrightarrow^d$ " 54) where " $\varphi \longleftrightarrow^d \psi \equiv (\varphi \supset^d \psi) \wedge^d (\psi \supset^d \varphi)$ "
16   definition AllD ("_{-}^d _") where " $\forall^d x. \varphi \equiv \neg^d(\exists^d x. \neg^d \varphi)$ "
17   definition AllD2 ("_{-}^d_2 _") where " $\forall^d_2 X. \varphi \equiv \neg^d(\exists^d_2 X. \neg^d \varphi)$ "
18   <Relative truth in a model. The first-order existential ranges over the individual domain D;
19   the second-order existential ranges over the collection E of admissible monadic sets.>
20   primrec RelativeTruthD::" $\mathcal{I} \Rightarrow \mathcal{D} \Rightarrow \mathcal{P} \Rightarrow \mathcal{E} \Rightarrow \mathcal{G} \Rightarrow \text{bool}$ " (" $\_^d\_^d\_^d\_^d\_^d$ " [10,10,10,10,10] 100) where
21     " $\langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d (r^d(x, y))$  =  $\text{I } r \ (g \ x) \ (g \ y)$ "
22     " $\langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d (X^d(x))$  =  $(G \ X) \ (g \ x)$ "
23     " $\langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d (\neg^d \varphi)$  =  $\neg \langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d \varphi$ "
24     " $\langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d (\varphi \wedge^d \psi)$  =  $(\langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d \varphi \wedge \langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d \psi)$ "
25     " $\langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d (\exists^d x. \varphi)$  =  $(\exists d:D. \langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g[x \leftarrow d], G \models^d \varphi)$ "
26     " $\langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d (\exists^d_2 X. \varphi)$  =  $(\exists S:E. \langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G[X \leftarrow S] \models^d \varphi)$ "
27   <Validity: true in every model, every domain-respecting first- and second-order assignment.>
28   definition ValD (" $\models^d$ " 9) where " $\models^d \varphi \equiv \forall \mathcal{I} \ \mathcal{D} \ \mathcal{E} \ g \ G. g \text{ into } \mathcal{D} \longrightarrow G \text{ into } \mathcal{E} \longrightarrow \langle \mathcal{I}, \mathcal{D}, \mathcal{E} \rangle, g, G \models^d \varphi$ "
29   <Auxiliary "full-domain" notion of validity: assignments range over the full types (Univ on D and on  $\mathcal{D} \Rightarrow \text{bool}$ ).>
30   definition ValD' (" $\models^{d'}$ " 9) where " $\models^{d'} \varphi \equiv \forall \mathcal{I} \ g \ G. \langle \mathcal{I}, \text{Univ}, \text{Univ} \rangle, g, G \models^d \varphi$ "
31   <General validity implies full-domain validity.>
32   lemma Val: " $\models^d \varphi \implies \models^{d'} \varphi$ " using ValD'_def ValD_def by simp
33   <Bag of definitions for collective unfolding.>
34   named_theorems DefD lemmas [DefD] = OrD_def ImpD_def IffD_def AllD_def AllD2_def ValD_def ValD'_def
35 end

```

Fig. 1. Deep embedding of MSO in HOL: the datatype $\mathcal{F}$ with the predication atom and the two binders, the derived connectives, the relative-truth predicate **RelativeTruthD**, and the validity definitions **ValD**, **ValD'**.

$\mathcal{I}$, $\mathcal{D}$, $\mathcal{E}$, g , G . The type of formulae is $\sigma := \mathcal{I} \Rightarrow \mathcal{D} \Rightarrow \mathcal{P} \Rightarrow \mathcal{E} \Rightarrow \mathcal{G} \Rightarrow \text{bool}$. Atoms consult the interpretation; the predication case applies the second-order assignment; the first-order existential pairs a meta-level HOL existential with a first-order assignment update; and the second-order existential pairs a meta-level HOL existential with a second-order assignment update. The pattern is identical to that of [12]; the only difference is that the formula type σ carries two assignments and two domain restrictions, because both the first-order (individual) and the second-order (set) quantifier must be accommodated. We write $\models^s$ for the resulting validity and $\models^{s'}$ for its full-domain variant ($\mathcal{D} = \text{Univ}$ and $\mathcal{E} = \text{Univ}$), the maximal-shallow analogues of $\models^d$ and $\models^{d'}$.

4.4 Minimal shallow embedding (locale-based)

The maximal-shallow type carries every dependency as an explicit argument, threaded throughout because each quantifier *updates* an assignment in its recursive call ($g[x \leftarrow d]$ for $\exists x$, $G[X \leftarrow S]$ for $\exists X$). The minimal embedding avoids this: each MSO existential is encoded by HOL's own binder over the matching variable namespace (V , V_2), and atoms read their variables' denotations from

```

1 theory MSOinHOL_shallow imports MSOinHOL_preliminaries
2 begin
3 <Maximal (heavyweight) shallow embedding of MSO in HOL; MSO formulas are HOL terms of the following type σ.>
4 type_synonym σ = "I ⇒ D ⇒ P ⇒ E ⇒ G ⇒ bool"
5 <The six primitive cases.>
6 definition AtmS :: "R ⇒ V ⇒ V ⇒ σ" ("s¹(·,·)") where "r¹(x,y) ≡ λ I D E g G. I r (g x) (g y)"
7 definition PrdS :: "V2 ⇒ V ⇒ σ" ("s¹(·)") where "x¹(x) ≡ λ I D E g G. (G X) (g x)"
8 definition NegS :: "σ ⇒ σ" ("¬s¹") [58]59 where "¬s¹φ ≡ λ I D E g G. ¬(φ I D E g G)"
9 definition AndS :: "σ ⇒ σ ⇒ σ" (infixr "∧s¹" 56) where "φ ∧s¹ ψ ≡ λ I D E g G. φ I D E g G ∧ ψ I D E g G"
10 definition ExS :: "V ⇒ σ ⇒ σ" ("∃s¹") 53 where "∃s¹x. φ ≡ λ I D E g G. ∃d:D. φ I D E (g[x←d]) G"
11 definition ExS2 :: "V2 ⇒ σ ⇒ σ" ("∃s¹₂") 53 where "∃s¹₂X. φ ≡ λ I D E g G. ∃S:E. φ I D E g (G(X←S))"
12 <Derived connectives.>
13 definition OrS :: "σ ⇒ σ ⇒ σ" (infixr "∨s¹" 54) where "φ ∨s¹ ψ ≡ ¬s¹(¬s¹φ ∧s¹ ¬s¹ψ)"
14 definition ImpS :: "σ ⇒ σ ⇒ σ" (infixr "⊃s¹" 55) where "φ ⊃s¹ ψ ≡ ¬s¹φ ∨s¹ ψ"
15 definition IffS :: "σ ⇒ σ ⇒ σ" (infixr "↔s¹" 54) where "φ ↔s¹ ψ ≡ (φ ⊃s¹ ψ) ∧s¹ (ψ ⊃s¹ φ)"
16 definition AllS :: "V ⇒ σ ⇒ σ" ("∀s¹") 53 where "∀s¹x. φ ≡ ¬s¹(∃s¹x. ¬s¹φ)"
17 definition AllS2 :: "V2 ⇒ σ ⇒ σ" ("∀s¹₂") 53 where "∀s¹₂X. φ ≡ ¬s¹(∃s¹₂X. ¬s¹φ)"
18 <Relative truth and validity (mirroring the deep embedding).>
19 definition RelTruthS :: "I ⇒ D ⇒ P ⇒ E ⇒ G ⇒ σ ⇒ bool" ("(·,·,·,·,·) ⊨s¹") [100,0,0,0,0] 100
20 where "(I,D,E),g,G ⊨s¹ φ ≡ φ I D E g G"
21 definition ValS :: "σ ⇒ bool" ("⊨s¹") 9 where "⊨s¹ φ ≡ ∀ I D E g G. g into D ⟶ G into E ⟶ (I,D,E),g,G ⊨s¹ φ"
22 <Auxiliary "full-domain" notion of validity: assignments range over the full types.>
23 definition ValS' ("⊨s¹'") 9 where "⊨s¹' φ ≡ ∀ I g G. (I,Univ,Univ),g,G ⊨s¹ φ"
24 <General validity implies full-domain validity.>
25 lemma Val_s: "⊨s¹ φ ⟹ ⊨s¹' φ" using ValS'_def ValS_def by simp
26 <Bag of definitions.>
27 named_theorems DefS lemmas DefS_defs [DefS] = AtmS_def PrdS_def NegS_def AndS_def ExS_def ExS2_def OrS_def
28 ImpS_def IffS_def AllS_def AllS2_def RelTruthS_def ValS_def ValS'_def
29 end
    
```

Fig. 2. Maximal shallow embedding (MSOinHOL_shallow.thy): the formula type σ , the six primitive cases (the two existentials thread g and G respectively), the derived connectives, and validity.

the assignment $(X^m(x) \text{ is } (GG X)(gg x))$, so a bound variable is a symbol looked up in the assignment rather than a witness installed into an updated one. The recursion therefore never changes II , gg , or GG ; this, with the absence of any world dependency, lets us promote them to locale parameters, and the formula type reduces to *bool*.

We package the minimal embedding as an Isabelle/HOL locale *MinS* [2] (theory `MSOinHOL_shallow_minimal_locale.thy`, Fig. 3), fixing (II, gg, GG) . The six primitive cases are *AtmM*, *PrdM* (predication $X^m(x)$ defined as $(GG X)(gg x)$), *NegM*, *AndM*, the first-order binder *ExM* over V , and the second-order binder *ExM2* over V_2 . As the formula type is *bool*, relative truth and validity *ValM* (written $\models^m$) are the identity. Both binders are declared as HOL *binders*, so that *sledgehammer* treats second-order MSO goals as ordinary HOL goals: bound first- and second-order variables become HOL-level bound variables of types V and V_2 , and the external provers reason about them directly without unfolding the deep-embedding syntax.

For practical experimentation the locale is also instantiated once at the global level (theory `MSOinHOL_shallow_minimal.thy`, Fig. 7): three uninterpreted constants *II*, *gg*, *GG* are introduced and a *global_interpretation* re-issues the locale notation at the global level. The locale formulation remains the single source of all theorems; the global interpretation propagates them to the constants without proof duplication.

```

1 theory MSOinHOL_shallow_minimal_locale imports MSOinHOL_preliminaries
2 begin
3 —<Minimal (lightweight) shallow embedding of MSO in HOL, packaged as a locale.
4 Since MSO carries no world dependency, the formula type collapses to <bool>.>
5 locale MinS = fixes II ::  $\mathcal{I}$  and gg ::  $\mathcal{E}$  and GG ::  $\mathcal{G}$ 
6 begin
7 —<Six primitive cases. ExM, ExM2 are HOL binders over the symbol types V, V2;
8 atoms consult II via gg, and membership consults GG via gg.>
9 definition AtmM :: "R  $\Rightarrow$  V  $\Rightarrow$  bool" (("_m'(_)'") where "rm(x,y)  $\equiv$  II r (gg x) (gg y)"
10 definition PrdM :: "V2  $\Rightarrow$  V  $\Rightarrow$  bool" (("_m'(_)'") where "Xm(x)  $\equiv$  (GG X) (gg x)"
11 definition NegM :: "bool  $\Rightarrow$  bool" ("¬m_" [58]59) where "¬mφ  $\equiv$  ¬φ"
12 definition AndM :: "bool  $\Rightarrow$  bool  $\Rightarrow$  bool" (infixr "∧m" 56) where "φ ∧m ψ  $\equiv$  φ ∧ ψ"
13 definition ExM :: "(V  $\Rightarrow$  bool)  $\Rightarrow$  bool" (binder "∃m" 53) where "∃md. φ d  $\equiv$  ∃d. φ d"
14 definition ExM2 :: "(V2  $\Rightarrow$  bool)  $\Rightarrow$  bool" (binder "∃m2" 53) where "∃m2D. φ D  $\equiv$  ∃D. φ D"
15 —<Derived connectives.>
16 definition OrM :: "bool  $\Rightarrow$  bool  $\Rightarrow$  bool" (infixr "∨m" 54) where "φ ∨m ψ  $\equiv$  ¬m(¬mφ ∧m ¬mψ)"
17 definition ImpM :: "bool  $\Rightarrow$  bool  $\Rightarrow$  bool" (infixr "⊃m" 55) where "φ ⊃m ψ  $\equiv$  ¬mφ ∨m ψ"
18 definition IffM :: "bool  $\Rightarrow$  bool  $\Rightarrow$  bool" (infixr "⟷m" 54) where "φ ⟷m ψ  $\equiv$  (φ ⊃m ψ) ∧m (ψ ⊃m φ)"
19 definition AllM :: "(V  $\Rightarrow$  bool)  $\Rightarrow$  bool" (binder "∀m" 53) where "∀md. φ d  $\equiv$  ∀d. φ d"
20 definition AllM2 :: "(V2  $\Rightarrow$  bool)  $\Rightarrow$  bool" (binder "∀m2" 53) where "∀m2D. φ D  $\equiv$  ∀D. φ D"
21 —<Relative truth and validity. As the formula type is <bool>, validity is the identity.>
22 definition ValM :: "bool  $\Rightarrow$  bool" ("⊨m_" 9) where "⊨m φ  $\equiv$  φ"
23 —<Bag of definitions.>
24 named_theorems DefM lemmas DefM_defs [DefM] = AtmM_def PrdM_def NegM_def AndM_def ExM_def ExM2_def
25 OrM_def ImpM_def IffM_def AllM_def AllM2_def ValM_def
26 end
27 end

```

Fig. 3. The locale `MinS` (`MSOinHOL_shallow_minimal_locale.thy`) packaging the minimal shallow embedding; parameters II , gg , GG ; the six primitive cases including the HOL binders `ExM` (over V) and `ExM2` (over V_2); the derived connectives; and validity.

Maximal vs. minimal: the trade-off. The two shallow embeddings encode the same object logic but package the semantic parameters oppositely. The *maximal* embedding keeps the interpretation and both assignments explicit in each translated formula (type σ): the parameters are under direct control and the translation lines up term-for-term with the deep embedding, which makes it the natural *faithfulness bridge* (FaithfulSD) and the place to express the full-domain standard readings $\models^s$, $\models^{s'}$; the price is weight, as every subterm threads I, D, E, g, G . The *minimal* embedding hides those parameters in the locale `MinS`, so a translated formula becomes an *ordinary HOL proposition* of type `bool` with each MSO binder a native HOL binder: `sledgehammer` and `nitpick` act on it directly, which makes it the convenient vehicle for downstream reasoning and for comparing classes of interpretations — general vs. standard (§7) — at the cost of leaving the semantic parameters implicit.

5 Two-sorted substitution machinery

Substitution lemmas are needed for the `deep`↔`minimal`-shallow faithfulness proof, to handle the two existential-quantifier cases. Since MSO has two variable namespaces, we develop *two* parallel substitution apparatuses, *Part A* (first-order) and *Part B* (second-order), in theory `MSOinHOL_deep_subst_lemma.thy` (Figs. 9–11). We use named variables and explicit capture-avoiding substitution

rather than a nominal or de Bruijn infrastructure, keeping the two namespaces and their mutual transparency visible in the formal text.

The two apparatuses. Each part provides the standard components — free/bound/fresh predicates, a variable-for-variable substitution `Subst` (written $[x \leftarrow z](\varphi)$; second-order `Subst2`), a substitutability predicate, the `SubstitutionLemma` (resp. `SubstitutionLemma2`, whose semantic side is the update $G(X \leftarrow G Z)$), and an α -renaming `ren_for_subst` composing into the safe substitution `ren_subst`, written $[x \leftarrow_r z](\varphi)$. The size-based induction principles `SInduct` and `QInduct` recurse on *both* binders. The outcomes used downstream are L27/N27 (safe substitution commutes with assignment update) and L29/N29, which re-express a deep existential as a HOL existential over a fresh symbol — exactly the equation the minimal translation needs.

Transparency across namespaces. A binder of one sort contributes *nothing* to the machinery of the other: binding no variable of that namespace, it descends through every predicate, substitution, renaming, and induction by a single uniform clause, structurally identical to the negation clause and dispatched by the same `auto/simp`. This holds *twice over* — the second-order binder `ExD2` is transparent for Part A and the first-order `ExD` for Part B — the only genuinely new leaf being the predication atom $X(x)$, which contributes a free first-order occurrence (x) to Part A and a free second-order one (X) to Part B. The α -renaming is what keeps substitution sound across a binder: replacing x by z in $\exists^d z. r^d(x, z)$ would *capture* the free z , which `ren_subst` prevents by first renaming the bound z to a fresh symbol. We additionally isolate the single-binder semantic core as `rename_eval/rename_eval2` (theory `MSOinHOL_subst_extras.thy`, Fig. 12).

6 Faithfulness, automated

An embedding is *faithful* when validity transfers in both directions — a formula is valid in MSO exactly when its HOL translation is — so the embedding neither loses nor invents theorems; we establish this, automatically, for all three.

Two mappings translate deep formulae into shallow ones (theory `MSOinHOL_faithfulness_locale.thy`, Fig. 4): `DpToShS` (deep to maximal-shallow), written $\llbracket \cdot \rrbracket$, and `DpToShM` (deep to minimal-shallow), written $\langle \cdot \rangle$. The first recurses transparently through every primitive, since both source and target use named-variable binders for both sorts. The second, declared inside the locale `MinS`, bridges the two named-variable binders of the deep embedding to the two HOL binders of the minimal embedding: its first-order existential clause inserts a first-order safe substitution $[v \leftarrow_r d](\varphi)$, and its second-order existential clause inserts a second-order safe substitution $[V \leftarrow_{r2} D](\varphi)$.

Pointwise and validity-level theorems. The $\text{deep} \leftrightarrow \text{maximal-shallow}$ correspondence `FaithfulSDlem` is derived by a single induction on φ with `auto simp: DefSDefD`; its validity-level form `FaithfulSD` — the statement for closed

validity rather than for a fixed model and assignment – then follows by unfolding the two validity definitions `ValD_def` and `ValS_def`. The `deep↔minimal` correspondence is first shown pointwise, `FaithfulMDlem`, by a single `induct` with rule `QInduct: simp add: DefDDefM` discharges the propositional and atomic cases, and the two existential cases are closed by `blast` after the safe-substitution lemmata `L27` and `N27` rewrite the translated binders. As the minimal validity `ValM` is the identity, `FaithfulMD` coincides with `FaithfulMDlem`, and `FaithfulMS` follows by composition with `FaithfulSDlem`.

Faithfulness across all locale interpretations. The locale form admits the all-interpretations theorem `FaithfulMS_all`: a formula holds in the minimal embedding under *every* interpretation (II, gg, GG) of `MinS` iff it holds, in the deep embedding, in every model whose first- and second-order domains are exactly the ranges of the assignments,

$$(\forall II\ gg\ GG. \models^m (\llbracket \varphi \rrbracket)) = (\forall I\ g\ G. \langle I, \text{Range } g, \text{Range } G \rangle, g, G \models^d \varphi).$$

The proof is a single `simp` with `FaithfulMD`. The forward direction toward full deep validity, `Deep_to_MinS`, holds outright: a deep-valid formula transfers to every minimal interpretation. The converse is exactly the surjectivity problem — whether the countable assignment ranges can be made to exhaust the full second-order domain — to which we now turn.

What FaithfulMS_all buys. `FaithfulMS_all` makes the minimal embedding usable as a *proof* device, not merely a definitional convenience. To establish a deep validity one discharges the corresponding goal in the minimal embedding — a plain HOL formula with two ordinary binders, on which `sledgehammer` and `auto` operate directly — and the theorem transports the result to the deep syntax over the range-determined model class; conversely, any `nitpick` countermodel against the minimal embedding refutes the deep formula. The only gap is the passage from the range-determined model class to the full general (Henkin-style) class, which §7 closes; for the standard reading it shows that unrestricted all-interpretations faithfulness is impossible, while an elementary-substructure restriction recovers it exactly.

7 The surjectivity problem, second-order

In the minimal-shallow embedding the first-order binder ranges over the namespace $V = \text{nat}$ through $gg: V \rightarrow D$, with effective range $\text{Range } gg$; the second-order binder ranges over $V_2 = \text{nat}$ through $GG: V_2 \rightarrow (D \Rightarrow \text{bool})$ — the type of monadic sets (predicates) over individuals — with effective range $\text{Range } GG$. Both ranges are countable, whereas the deep embedding lets the existentials range over the full domains D and E , so faithfulness against the full domains does not follow from `FaithfulMD`. By `FaithfulMS_all` the missing converse is precisely

$$(\forall I\ g\ G. \langle I, \text{Range } g, \text{Range } G \rangle, g, G \models^d \varphi) \implies \models^d \varphi.$$

```

1 theory MSOinHOL_faithfulness_locale
2 imports MSOinHOL_deep MSOinHOL_shallow MSOinHOL_shallow_minimal_locale MSOinHOL_deep_subst_lemma
3 begin
4   <Translation from deep to maximal shallow (transparent: both use named-variable binders).>
5   fun DpToShS ("[" "]") where "[rd(x,y)] = rs(x,y)" | "[Xd(x)] = Xs(x)" | "[¬dφ] = ¬s[φ]" | "[φ ∧d ψ] = [φ] ∧s [ψ]"
6   | "[∃dv. φ] = (∃sv. [φ])" | "[∃d2v. φ] = (∃s2v. [φ])"
7   <Translation from deep to minimal shallow (within the locale MinS): each binder is bridged by safe substitution
8   into a HOL-level binder -- first-order via <[v ← d]>, second-order via <[V ← r2D]>.>
9   fun (in MinS) DpToShM ("[" "]") where "[rd(x,y)] = rm(x,y)" | "[Xd(x)] = Xm(x)" | "[¬dφ] = ¬m[φ]" | "[φ ∧d ψ] = [φ] ∧m [ψ]"
10  | "[∃dv. φ] = (∃md. ([v ← d](φ)))" | "[∃d2v. φ] = (∃m2D. ([V ← r2D](φ)))"
11  <Faithfulness deep ↔ maximal shallow. Pointwise first; global by unfolding validity.>
12  theorem FaithfulSDlem: "(⟦(I,D,E),g,G ⊨s [φ]⟧ ↔ (⟦(I,D,E),g,G ⊨d φ⟧)" by (induct φ arbitrary: g G; auto simp: DefS DefD)
13  theorem FaithfulSD: "(⊨s [φ]) ↔ (⊨d φ)" by (simp add: FaithfulSDlem ValD_def ValS_def)
14  <Faithfulness deep ↔ minimal shallow, parametrised by the locale, relative to the ranges of gg and GG.>
15  context MinS begin
16    theorem FaithfulMDlem: "(⟦(I,Range gg,Range GG),gg,GG ⊨d φ⟧"
17    by (induct φ rule: QInduct; simp add: DefD DefM) blast+
18    theorem FaithfulMD: "(⊨m [φ]) ↔ (⟦(I,Range gg,Range GG),gg,GG ⊨d φ⟧"
19    using FaithfulMDlem by (simp add: ValM_def)
20    theorem FaithfulMSlem: "(⟦(I,Range gg,Range GG),gg,GG ⊨s [φ]⟧"
21    using FaithfulSDlem FaithfulMDlem by auto
22    theorem FaithfulMS: "(⊨m [φ]) ↔ (⟦(I,Range gg,Range GG),gg,GG ⊨s [φ]⟧"
23    using FaithfulMSlem by (simp add: ValM_def)
24  end
25  <Global faithfulness: a formula holds in every minimal interpretation of MinS iff it holds in every model
26  whose first- and second-order domains are exactly the ranges of the chosen assignments.>
27  theorem FaithfulMS_all: "(∀(I gg GG. MinS.ValM (MinS.DpToShM II gg GG φ)) = (∀(g G. (I,Range g,Range G),g,G ⊨d φ))"
28  by (simp add: MinS.FaithfulMD)
29  <One direction toward full deep validity: deep validity transfers to every minimal interpretation.
30  The converse is the (second-order) surjectivity problem discussed in the paper.>
31  theorem Deep_to_MinS: "(⊨d φ) ⇒ (∀(I gg GG. MinS.ValM (MinS.DpToShM II gg GG φ))"
32  by (metis (mono_tags, lifting) MinS.FaithfulMD ValD_def)
33  <Consistency check.>
34  lemma True nitpick[satisfy] oops
35 end

```

Fig. 4. Faithfulness: the translations $\llbracket \cdot \rrbracket$ and $\llbracket \cdot \rrbracket$ (the latter bridging *both* binders via the two safe substitutions), the deep $\leftrightarrow$ maximal-shallow theorems, the locale-based deep $\leftrightarrow$ minimal-shallow correspondences, the all-interpretations theorem FaithfulMS_all, and the one-directional bridge Deep_to_MinS.

The decisive observation is that the general validity $\models^d$ imposes *no* closure on the admissible collection E : it quantifies over arbitrary families with G into E . That is exactly the *general* (Henkin-style) reading of MSO,⁴ which is nothing but two-sorted first-order logic over the structure with individuals as one sort,

⁴ The reading $\models^d$ is the class of structures $\langle I, D, E \rangle$ with an *arbitrary* admissible family $E \subseteq \mathcal{P}(D)$ and no closure condition; terminology for it is not uniform. Shapiro [21] calls these *Henkin models*, reserving *faithful Henkin model* for one that additionally satisfies comprehension and choice, and *full model* for the standard case $E = \mathcal{P}(D)$. Henkin [18], Väänänen [23], and Manzano [19] instead call the comprehension-closed structures *Henkin models* and the unrestricted ones *general models*. We write “general (Henkin-style)” for $\models^d$ to stay neutral: it is Shapiro’s *Henkin* class and Väänänen’s *general* class. On either convention the levels nest as full/standard $\subseteq$ comprehension-closed (*faithful Henkin/Henkin*) $\subseteq$ general, with $\models^d$ the widest; the diagonal comprehension instance separates $\models^d$ from the comprehension-closed level, while the categoricity phenomenon separates that level from the full one. Via the reduction of this semantics to many-sorted first-order logic (Shapiro [21, Thm. 3.5]; Manzano [19]), $\models^d$ inherits the downward Löwenheim–Skolem theorem directly.

the admissible sets as a second sort, the relation symbols as binary relations on the first sort, and a membership relation $d \in S \Leftrightarrow S(d)$ between the sorts. First-order logic has the downward Löwenheim–Skolem property; the apparent “second-order” difficulty is an artefact of conflating $\models^d$ with the *standard* reading $\models^{d'}$ treated below.

The converse (range-relative $\Rightarrow$ deep validity) holds, for the general reading. Together with Deep.to.MinS, this gives the following exact characterisation of range-relative validity.

Theorem 1. *For every formula φ , range-relative validity coincides with general (Henkin-style) deep validity:*

$$(\forall I g G. \langle I, \text{Range } g, \text{Range } G \rangle, g, G \models^d \varphi) \iff \models^d \varphi.$$

Equivalently, via FaithfulMS_all, the minimal-shallow embedding is faithful exactly to general (Henkin-style) MSO validity.

The proof (App. A) is a two-sorted downward Löwenheim–Skolem construction. From a countermodel one extracts a *countable* elementary sub-model — one satisfying exactly the same formulae — containing the assignment ranges by the Tarski–Vaught test (a witness-reflection criterion that guarantees elementarity), drawing set-witnesses honestly from E — so that no comprehension is assumed and the sub-model stays inside the general class — and then reindexes the countably many variables to surject onto the countable sub-domains while fixing the finitely many free variables of φ , using a coincidence lemma. The second-order domain E is never replaced by a powerset; being “comprehension-closed” is a property the hull *enjoys*, not a hypothesis it requires.

Unrestricted minimal interpretations do not capture the standard reading. No Löwenheim–Skolem argument can strengthen this to the *standard* validity $\models^{d'}$ (where $E = \mathcal{P}(D)$), because the two notions genuinely differ. Writing $\models_r^d$ for range-relative validity — the left-hand side of Theorem 1, there identified with $\models^d$ — we have

$$\models_r^d = \models^d \subsetneq \models^{d'}.$$

The inclusion $\models^d \subseteq \models^{d'}$ is immediate — the standard model is itself one of the general models quantified over by $\models^d$, which is the content of the lemma we name Val — while its strictness is witnessed by comprehension: the instance $\exists X. \forall x. (X(x) \leftrightarrow P(x, x))$ is $\models^{d'}$ -valid (**comprehension_atom**) but not $\models^d$ -valid, as the **nitpick** countermodels of §8, over a restricted E that omits the diagonal set $\{d : P(d, d)\}$, show. In the three-level hierarchy of Footnote 4, this instance separates the *general* reading $\models^d$ from the comprehension-closed *Henkin* reading, and hence *a fortiori* from the full reading $\models^{d'}$; the residual gap, between the Henkin and the full reading, is the familiar *categoricity* phenomenon — the full reading pinning models down up to isomorphism — since monadic quantification over sets already expresses Dedekind completeness (quantify over the bounded set whose least upper bound is asserted), so a sentence forcing a complete dense

order without endpoints has only uncountable full models and admits no downward Löwenheim–Skolem under the standard reading. The machine-checked witness in our development is the simpler diagonal comprehension instance above.

The analysis thus resolves into a dichotomy: the converse is a *theorem* for the general (Henkin-style) reading and a provable *non-reduction* for the standard reading. The supplementary theories `MSOinHOL_lowenheim_skolem_lemmas.thy` and `MSOinHOL_lowenheim_skolem.thy` carry this out — the coincidence lemma `coincidence`, the surjective reindexing `reindex`, the assembled converse `RangeValid_imp_ValD` (with corollaries `RangeValid_iff_ValD` and `Faithful_to_Henkin`), the negative `Standard_strictly_stronger` with its explicit diagonal countermodel, and the two-sorted elementary-substructure hull `ls_hull`, a countable Tarski–Vaught hull `skolem_hull` whose truth-preservation lemma `truth_pres` drives the two-sorted construction — and, built on the same hull, the elementary-substructure relation `ElementarySubstructure`, the first-class downward Löwenheim–Skolem theorem `DownwardLowenheimSkolem`, and the standard-reading characterisation `Deep'_to_MinS`. All are complete structured proofs, machine-checked under Isabelle2025-2; the construction is given in App. A.

The standard reading, captured exactly by elementary interpretations. The non-reduction says only that no *range-relative* — equivalently, all-interpretations — faithfulness can reach $\models^{d'}$. It does not say the standard reading escapes the minimal embedding: it does not. Writing $\langle I', D', E' \rangle \subseteq_E \langle I, D, E \rangle$ for the two-sorted *elementary substructure* relation `ElementarySubstructure` ($D' \sqsubseteq D$, $E' \sqsubseteq E$, and the two models satisfy the same formulae under every assignment landing in the smaller pair), the development now isolates the Tarski–Vaught hull as a *first-class* downward Löwenheim–Skolem theorem `DownwardLowenheimSkolem`: every countable nonempty seed sub-pair of a model extends to a countable elementary substructure. Using it, the strongest faithfulness theorem `Deep'_to_MinS` characterises the standard reading *exactly*,

$$\begin{aligned} \models^{d'} \varphi &\iff \forall II \, gg \, GG. \\ &\langle II, \text{Range } gg, \text{Range } GG \rangle \subseteq_E \langle II, \text{Univ}, \text{Univ} \rangle \longrightarrow \models^m (\varphi), \end{aligned}$$

where $\langle II, \text{Univ}, \text{Univ} \rangle$ is the full model ($E = \mathcal{P}(D)$). Both readings of MSO are therefore recovered from the minimal embedding, differing only in the admissible interpretations: $\models^d$ quantifies over *all* minimal interpretations (Theorem 1, via `FaithfulMS_all`), whereas $\models^{d'}$ quantifies over the *elementary* ones — those whose countable range model is an elementary substructure of the full model. The strict inclusion $\models^d \subsetneq \models^{d'}$ is exactly the statement that this elementary restriction cannot be dropped, the diagonal comprehension instance witnessing it. Concretely, the `nitpick` countermodels to the closure schemas in the minimal embedding (§8) are necessarily *non-elementary* interpretations: their countable range omits a set the schema needs, so they refute it; on the elementary interpretations the same schemas become provable, in agreement with their $\models^{d'}$ -validity. (They are

thus not genuine counterexamples to the standard reading, but artefacts of admitting non-elementary ranges; under the elementary-substructure restriction (the locale `MinS_ES.Univ`) each would be a theorem.)

A recursion-theoretic strengthening. The non-reduction just established is model-theoretic: the failure of downward Löwenheim–Skolem shows that the full MSO models — those with $E = \mathcal{P}(D)$ — do not form an elementary class. A complementary, recursion-theoretic obstruction is that standard validity $\models^{d'}$ is not recursively enumerable — a classical consequence of the expressive strength of standard second-order logic (cf. Shapiro [21], Väänänen [23]), which we do not formalise here — so it is the consequence set of no recursively axiomatisable first-order theory. This non-enumerability reflects quantification over *all* interpretations of the relation symbols, not the standard reading as such: restricting to a fixed structure or class of structures — $(\mathbb{N}, <)$, the infinite binary tree, or finite words and trees — makes standard MSO decidable, hence complete and enumerable [14,22,16].

8 Experiments

We illustrate the development with two test files. The core file `MSOinHOL_experiments.thy` (Fig. 5) collects tautologies, comprehension, and `nitpick` disproofs; the larger file `MSOinHOL_experiments_classic.thy` replicates a suite of classical MSO landmarks across all three embeddings (§8.1). Propositional tautologies (proved in all three embeddings with the d/s/m discipline), first-order tautologies such as $(\forall x. P(x, x)) \supset (\exists x. P(x, x))$, and the predication tautology $\forall x. (X(x) \supset X(x))$ are dispatched by `simp` or `auto` after unfolding `DefD`, `DefS`, or `DefM`.

Monadic comprehension. The principal second-order validity we verify is monadic *comprehension*, stated over the full second-order domain ($\models^{d'}$, $E = \mathbf{Univ}$). In its general form it is a schema, parametric in an arbitrary formula φ in which the comprehension variable X does not occur free:

$$\models^{d'} \exists X. \forall x. (X(x) \leftrightarrow \varphi).$$

The witnessing set is exactly the property defined by φ , namely the HOL predicate $\lambda d. \langle I, \mathbf{Univ}, \mathbf{Univ} \rangle, g[x \leftarrow d], G \models^d \varphi$; it always lies in E because $E = \mathcal{P}(D)$ in the standard semantics, and the side condition on X lets the second-order irrelevance lemma discharge the X -slot inside φ . We record this as the theorem `comprehension_schema` (supplementary theory `MSOinHOL_comprehension.thy`, Fig. 13); the two instances proved directly in `MSOinHOL_experiments.thy` — the set of P -self-related individuals, $\exists X. \forall x. (X(x) \leftrightarrow P(x, x))$, with witness $\lambda d. P d d$, and the universally-true predicate $\exists X. \forall x. X(x)$, with witness $\lambda d. \mathbf{True}$ — are its corollaries. No second-order search is required, because the comprehension term is itself an ordinary HOL predicate.

Disproofs by nitpick. `nitpick` reliably finds finite countermodels for invalid schemata; these are exploratory searches, terminated with `oops` rather than stored as theorems. “Every monadic predicate holds somewhere”, $\forall X. \exists x. X(x)$, fails by the empty predicate; the comprehension instance asking for a predicate that both holds and fails of every individual, $\exists X. \forall x. (X(x) \wedge \neg X(x))$, is unsatisfiable hence invalid; and symmetry of P , $\forall x \forall y. (P(x, y) \supset P(y, x))$, fails by a two-element model. Each disproof uses the full-domain relation $\models^{d'}$.

```

1 theory MSOinHOL_experiments imports MSOinHOL_deep MSOinHOL_shallow MSOinHOL_shallow_minimal
2 begin
3   abbreviation "(x::V) ≡ 1" abbreviation "(y::V) ≡ 2" abbreviation "(z::V) ≡ 3"
4   abbreviation "(X::V2) ≡ 1" abbreviation "(Y::V2) ≡ 2"
5   consts P::R
6   <--- Propositional tautologies, lifted to MSO in all three embeddings --->
7   lemma "⊨d (Pd(x,x) ⊃d Pd(x,x))" unfolding DefD by simp
8   lemma "⊨s (Ps(x,x) ⊃s Ps(x,x))" unfolding DefS by simp
9   lemma "⊨m (Pm(x,x) ⊃m Pm(x,x))" unfolding DefM by simp
10  lemma "⊨d (¬d Pd(x,y) ⊃d Pd(x,y))" unfolding DefD by auto
11  lemma "⊨s (¬s Ps(x,y) ⊃s Ps(x,y))" unfolding DefS by auto
12  lemma "⊨m (¬m Pm(x,y) ⊃m Pm(x,y))" unfolding DefM by auto
13  <--- First-order tautologies (the individual domain is inhabited whenever some assignment exists) --->
14  lemma "⊨d (∀dx. Pd(x,x) ⊃d (∃dx. Pd(x,x)))" unfolding DefD by auto
15  lemma "⊨s (∀sx. Ps(x,x) ⊃s (∃sx. Ps(x,x)))" unfolding DefS by auto
16  lemma "⊨m (∀mx. Pm(x,x) ⊃m (∃mx. Pm(x,x)))" unfolding DefM by auto
17  <--- Membership tautology: every individual that is in X is in X --->
18  lemma "⊨d (∀dx. (Xd(x) ⊃d Xd(x)))" unfolding DefD by auto
19  lemma "⊨s (∀sx. (Xs(x) ⊃s Xs(x)))" unfolding DefS by auto
20  lemma "⊨m (∀mx. (Xm(x) ⊃m Xm(x)))" unfolding DefM by auto
21  <--- Monadic comprehension (headline second-order validity): the set of P-self-related individuals
22  exists. Stated over the full second-order domain, where every monadic set is admissible. --->
23  lemma comprehension_d: "⊨d (∃d2X. ∀dx. ((Xd(x) ↔d Pd(x,x)))" unfolding DefD apply (intro all; simp) by meson
24  lemma comprehension_s: "⊨s (∃s2X. ∀sx. ((Xs(x) ↔s Ps(x,x)))" unfolding DefS apply (intro all; simp) by meson
25  <--- A universal monadic set exists over the full domain (the predicate that holds everywhere) --->
26  lemma "⊨d (∃d2X. ∀dx. (Xd(x)))" unfolding DefD apply simp by (rule exI[where x="λd. True"]) simp
27  <--- Invalid schemata: nitpick reports finite countermodels (using the full-domain relation --->
28  <Not every monadic set is inhabited: the empty set is a countermodel to "∀X ∃x. x ∈ X".>
29  lemma "⊨d (∀d2X. ∃dx. (Xd(x)))" unfolding DefD apply simp nitpick oops
30  <Symmetry of P is not implied: nitpick produces a 2-element countermodel.>
31  lemma "⊨d (∀dx. ∀dy. (Pd(x,y) ⊃d Pd(y,x)))" unfolding DefD apply simp nitpick oops
32  <Comprehension fails to give a set disjoint from itself: "∃X ∀x. x ∈ X ∧ ¬x ∈ X" is unsatisfiable, hence invalid.>
33  lemma "⊨d (∃d2X. ∀dx. ((Xd(x) ∧d ¬d(Xd(x))))" unfolding DefD apply simp nitpick oops
34 end

```

Fig. 5. Experiments: propositional and first-order tautologies in all three embeddings, the predication tautology, monadic comprehension and the universal-predicate validity over the full second-order domain, and `nitpick` disproofs of “every predicate holds somewhere”, a self-contradictory predicate, and symmetry of P .

8.1 Classical MSO landmarks across the three embeddings

A second, larger experiment file (`MSOinHOL_experiments_classic.thy`, Fig. 14) revisits a sequence of textbook MSO constructions and runs each one through all three embeddings. The point is not the individual validities — they are folklore — but the *pattern* of their truth values across the five readings $\models^d$, $\models^{d'}$, $\models^s$, $\models^{s'}$, $\models^m$, which is the operational face of the standard-vs-general dichotomy of §7. Table 1 collects the outcomes.

Table 1. Classical MSO landmarks across the five readings. In the full-domain columns $\models^{d'}$, $\models^{s'}$ ($E = \mathbf{Univ}$) and the minimal column $\models^m$, $\checkmark$ is a validity proof (an explicit second-order witness for $\models^{d'}$ / $\models^{s'}$), $\times$ a **nitpick**-found countermodel, and “ $\neg$ ” a case not separately tested (failing as complement/intersection do). The general readings $\models^d$, $\models^s$ are not tested independently: they coincide with $\models^m$ by **FaithfulSD** ($\models^s = \models^d$) and Theorem 1 ($\models^d$ is the minimal embedding over all interpretations); the full readings likewise agree, $\models^{d'} = \models^{s'}$.

Schema (and witness, where valid)	$\models^d$	$\models^{d'}$	$\models^s$	$\models^{s'}$	$\models^m$
Complement $\lambda d. \neg S(d)$	$\times$	$\checkmark$	$\times$	$\checkmark$	$\times$
Intersection $\lambda d. S_1(d) \wedge S_2(d)$	$\times$	$\checkmark$	$\times$	$\checkmark$	$\times$
Union $\lambda d. S_1(d) \vee S_2(d)$	$\neg$	$\checkmark$	$\neg$	$\checkmark$	$\neg$
Separation $\lambda d. S(d) \wedge I(P)(d, d)$	$\neg$	$\checkmark$	$\neg$	$\checkmark$	$\neg$
P -image $\lambda d. \exists e. S(e) \wedge I(P)(e, d)$	$\neg$	$\checkmark$	$\neg$	$\checkmark$	$\neg$
P -preimage $\lambda d. \exists e. I(P)(d, e) \wedge S(e)$	$\neg$	$\checkmark$	$\neg$	$\checkmark$	$\neg$
Reachability $x \rightsquigarrow y$	$\times$	$\times$	$\times$	$\times$	$\times$
Reflexive reachability $x \rightsquigarrow x$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
2-colorability	$\times$	$\times$	$\times$	$\times$	$\times$

Boolean closure and graph operations. The first six rows are instances of the comprehension schema above: each asserts the set defined by a formula φ , i.e. $\exists X. \forall d. (X(d) \leftrightarrow \varphi(d))$, with witness $\lambda d. \varphi(d)$ (the predicate shown in the table). Over the full second-order domain ($\models^{d'} / \models^{s'}$, $E = \mathcal{P}(D)$) each witness is an ordinary HOL predicate supplied by a single **exI**: the Boolean cases [14,22] use $\lambda d. \neg S(d)$, $\lambda d. S_1(d) \wedge S_2(d)$, $\lambda d. S_1(d) \vee S_2(d)$, and the monadic graph operations [16] the witnesses shown in the table. After unfolding **DefD** (resp. **DefS**) and β -reducing, the deep and maximal-shallow goals are HOL-identical, so the two proof scripts are line-for-line the same — the d/s alignment made visible on non-trivial second-order content. In the minimal embedding $\exists_2^m$ ranges over $V_2 = \text{nat}$ through GG , so the required predicate need not lie in $\text{Range } GG$ and **nitpick** refutes *complement* and *intersection* (the remaining four fail for the same reason).

Reachability and colorability. Reachability and 2-colorability are genuinely invalid in MSO, independently of the standard/general choice, and fail in *all* columns: reachability [4] — “if Z contains x and is closed under P -successors then Z contains y ” — has a **nitpick** countermodel with x, y P -disconnected; its reflexive variant (conclusion $Z(x)$) follows from the first conjunct alone, needs no witness, and so survives even the minimal embedding; 2-colorability [22] fails on an odd cycle (K_3). The overall pattern — closure rows valid under the full readings $\models^{d'} / \models^{s'}$ but refuted under the general $\models^d / \models^s / \models^m$, reachability and 2-colorability uniformly refuted — is exactly the inclusion $\models^d \subsetneq \models^{d'}$ of §7 read off machine-checked outcomes: faithful to the *general* (Henkin-style) reading, the minimal embedding refutes precisely the principles that separate the standard reading from the general one, while reflexive reachability, needing no set witness, holds across the board.

Locale-based transfer. A short demonstration (theory `MSOinHOL_experiments_locale.thy`, Fig. 15) proves a schema purely from minimal-shallow definitions, inside the locale, and transfers it to the range-relative deep embedding by a single application of `MinS_to_Deep`, the directional reading of `FaithfulMS_all`. This is the operational pay-off of the locale design: the user proves once in the simplest setting, and the strong faithfulness theorem ports the result to the syntactic deep embedding.

9 Conclusion

`MSOinHOL` demonstrates that the deep-and-shallow methodology of [12] scales to a setting with *two* binders over different sorts. The new ingredients are a two-sorted substitution apparatus, in which each binder is transparent for the other namespace, and a locale-based minimal embedding with two HOL binders and an all-interpretations faithfulness theorem. The powerset character of the second-order domain turns the surjectivity problem into a clean dichotomy. On one side, range-relative validity coincides with the general (Henkin-style) reading of MSO — a two-sorted Löwenheim–Skolem theorem (§7). On the other, the standard reading is provably stronger, yet it too is captured exactly by the minimal embedding once one restricts to *elementary* interpretations (`Deep'_to.MinS`). The two readings are thus one and the same translation, differing only in the class of interpretations over which validity is quantified. The whole construction — coincidence, the two-sorted elementary-substructure hull, reindexing, the assembled converse, and the negative result — is machine-checked. A suite of classical MSO landmarks — Boolean closure, graph operations, reachability, 2-colorability — replicated across the three embeddings turns this dichotomy into a table of validity proofs, `nitpick`-found countermodels, and entailed cases: the closure principles are valid under the full second-order domain and refuted by the general minimal embedding.

The ingredients are individually classical: downward Löwenheim–Skolem has been formalised in Isabelle for first-order logic [13]; the distinction between the *standard* and the *general* (Henkin-style) reading of second-order semantics is textbook [23]; and the general reading is known to be, in effect, a two-sorted first-order theory [19]. What is new here, to our knowledge, is to carry a mechanised downward Löwenheim–Skolem theorem all the way through to *monadic second-order* validity. The construction is a two-sorted elementary-substructure argument that, at each step, takes the set it needs as a witness from the model’s own admissible collection E rather than forming it as an arbitrary subset — and that is exactly what keeps the result inside the general (Henkin-style) class, where such sets need not exist in advance.

Use in teaching. Beyond its metatheoretic interest, the development doubles as teaching material for logic at the graduate and even undergraduate level. Because the shallow embeddings turn MSO formulae into ordinary Isabelle/HOL goals, the precise difference between the standard and the general (Henkin-style) reading — and its consequences for MSO — becomes something students

can explore hands-on and around the clock, through interactive proof and with `sledgehammer` and `nitpick`. A principle that holds under the full second-order domain yet fails in the minimal, range-restricted embedding makes the standard-vs-general gap immediately tangible: a machine-checked proof in one column and an explicit countermodel in another.

Availability. The full Isabelle/HOL development is published as the Archive of Formal Proofs entry `MSOinHOL` [6]: seventeen theory files (the ten core theories together with seven supplementary ones — `subst.extras`, `comprehension`, `experiments.classic`, the elementary minimal interpretation `shallow_minimal_elementary` and its experiments `experiments.classic.elementary`, and the two-sorted Löwenheim–Skolem development split into `lowenheim.skolem.lemmas` and `lowenheim.skolem`), loaded in dependency order by the session `ROOT`; the published entry adds one further experiments theory (`experiments.extra`) not required for the results reported here. The development has been checked in Isabelle2025-2 on a MacBook Air (Apple M3, 16 GB). An extended version of this paper, with the full two-sorted Löwenheim–Skolem proof and the rendered Isabelle/HOL theory sources as appendices, is available on arXiv [7].

Disclosure on the use of generative AI. The authors used generative AI assistants (Anthropic’s Claude family of models) to draft and shorten prose, to propose and shorten some proofs (in particular parts of the two-sorted Löwenheim–Skolem construction), and to maintain cross-references between the LaTeX source and the Isabelle/HOL files. The mathematical content, the theory development, and the design choices are the authors’; all text and proofs in the final manuscript have been verified by the authors, who take full responsibility for the content.

References

1. Ayari, A., Basin, D.: Bounded model construction for monadic second-order logics. In: Computer Aided Verification (CAV 2000). Lecture Notes in Computer Science, vol. 1855, pp. 99–112. Springer (2000)
2. Ballarín, C.: Locales: A module system for mathematical theories. Journal of Automated Reasoning **52**(2), 123–153 (2014). <https://doi.org/10.1007/s10817-013-9284-7>
3. Basin, D., Klarlund, N.: Automata based symbolic reasoning in hardware verification. Formal Methods in System Design **13**(3), 255–288 (1998)
4. Basin, D.A., Klarlund, N.: Hardware verification using monadic second-order logic. In: Computer Aided Verification (CAV 1995). Lecture Notes in Computer Science, vol. 939. Springer (1995). https://doi.org/10.1007/3-540-60045-0_38
5. Benz Müller, C., Brown, C., Kohlhasse, M.: Higher-order semantics and extensionality. Journal of Symbolic Logic **69**(4), 1027–1088 (2004). <https://doi.org/10.2178/jsl/1102022211>
6. Benz Müller, C., Kirchner, D.: Monadic second-order logic in HOL: Deep and shallow embeddings with automated faithfulness. Archive of Formal Proofs (2026), formal proof development, <https://isa-afp.org/entries/MSOinHOL.html>

7. Benz Müller, C., Kirchner, D.: Monadic second-order logic in HOL: Deep and shallow with automated faithfulness (extended preprint). arXiv preprint arXiv:XXXX.XXXXX (2026), extended version, with the full Löwenheim–Skolem proof and the rendered Isabelle/HOL theory sources as appendices
8. Benz Müller, C., Kirchner, D., Pasetto, L.: Many logics, one methodology: A plea for logical pluralism in formalised reasoning. arXiv preprint arXiv:2605.27246 [cs.LO] (2026)
9. Benz Müller, C., Parent, X., van der Torre, L.: Designing normative theories for ethical and legal reasoning: LogiKEy framework, methodology, and tool support. *Artificial Intelligence* **287**, 103348 (2020). <https://doi.org/10.1016/j.artint.2020.103348>
10. Benz Müller, C., Paulson, L.C.: Multimodal and intuitionistic logics in simple type theory. *The Logic Journal of the IGPL* **18**(6), 881–892 (2010). <https://doi.org/10.1093/jigpal/jzp080>
11. Benz Müller, C., Paulson, L.C.: Quantified multimodal logics in simple type theory. *Logica Universalis (Special Issue on Multimodal Logics)* **7**(1), 7–20 (2013). <https://doi.org/10.1007/s11787-012-0052-y>
12. Benz Müller, C.: Faithful logic embeddings in HOL — deep and shallow. In: Barrett, C., Waldmann, U. (eds.) *Automated Deduction – CADE-30 – 30th International Conference on Automated Deduction*, Stuttgart, Germany, July 28–31, 2025, *Proceedings. Lecture Notes in Computer Science*, vol. 15943, pp. 280–302. Springer (2025). https://doi.org/10.1007/978-3-031-99984-0_16, (preprint: arXiv:2502.19311)
13. Berghofer, S.: First-order logic according to fitting. *Archive of Formal Proofs* (2007), formal proof development, includes a mechanised downward Löwenheim–Skolem theorem for countable first-order logic, <https://isa-afp.org/entries/FOL-Fitting.html>
14. Büchi, J.R.: Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **6**(1–6), 66–92 (1960)
15. Church, A.: A formulation of the simple theory of types. *Journal of Symbolic Logic* **5**(2), 56–68 (1940). <https://doi.org/10.2307/2266170>
16. Courcelle, B., Engelfriet, J.: *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*, *Encyclopedia of Mathematics and its Applications*, vol. 138. Cambridge University Press (2012)
17. van Heijenoort, J.: *From Frege to Gödel: A Source Book in Mathematics, 1879–1931*. Source books in the history of the sciences series, Harvard Univ. Press, Cambridge, MA, 3rd printing, 1997 edn. (1967)
18. Henkin, L.: Completeness in the theory of types. *Journal of Symbolic Logic* **15**(2), 81–91 (1950). <https://doi.org/10.2307/2266967>
19. Manzano, M.: *Extensions of First-Order Logic*, *Cambridge Tracts in Theoretical Computer Science*, vol. 19. Cambridge University Press (1996)
20. Nipkow, T., Paulson, L.C., Wenzel, M.: *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, *Lecture Notes in Computer Science*, vol. 2283. Springer (2002)
21. Shapiro, S.: *Foundations without Foundationalism: A Case for Second-Order Logic*, *Oxford Logic Guides*, vol. 17. Oxford University Press (1991)
22. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) *Handbook of Formal Languages*, Vol. 3: Beyond Words, pp. 389–455. Springer (1997)
23. Väänänen, J.: *Models and Games*, *Cambridge Studies in Advanced Mathematics*, vol. 132. Cambridge University Press (2011)

A The converse of FaithfulMS_all: range vs. general (Henkin-style) validity

We prove Theorem 1. Write $\models_r^d \varphi$ for range-relative validity,

$$\models_r^d \varphi \equiv \forall I g G. \langle I, \text{Range } g, \text{Range } G \rangle, g, G \models^d \varphi.$$

The forward direction $\models^d \varphi \Rightarrow \models_r^d \varphi$ is immediate (it is `Deep.to.MinS`: g maps into $\text{Range } g$ and G into $\text{Range } G$, so both are admissible). We prove the converse.

Setup. A model $\langle I, D, E \rangle$ is read as a two-sorted first-order structure over the fixed individual type: sort 1 is $\{d : D\}$, sort 2 is $\{S : E\}$, each relation symbol r denotes the binary relation $I r$ on sort 1, and a single membership relation is defined by $\text{mem}(d, S) \Leftrightarrow S(d)$. Under this reading $\langle I, D, E \rangle, g, G \models^d \varphi$ is ordinary two-sorted FO satisfaction, with $\exists^d x$ ranging over sort 1 and $\exists_2^d X$ over sort 2; *no* closure on E is imposed.

Coincidence (generalising L12/N12). By induction on φ : if g, g' agree on every first-order variable free in φ and G, G' agree on every second-order variable free in φ , then $\langle I, D, E \rangle, g, G \models^d \varphi \Leftrightarrow \langle I, D, E \rangle, g', G' \models^d \varphi$. The binder cases use that an update $g[y \leftarrow d]$ (resp. $G\langle Y \leftarrow S \rangle$) preserves agreement on the free variables of the body.

Proof. Converse of Theorem 1, contrapositively. Suppose $\not\models^d \varphi$. Then there are I, D, E, g, G with g into D , G into E , and $\langle I, D, E \rangle, g, G \not\models^d \varphi$.

Step 1 (downward Löwenheim–Skolem). Build sub-predicates $D_0 \sqsubseteq D$ and $E_0 \sqsubseteq E$ by closing $\text{Range } g \cup \text{Range } G$ under Tarski–Vaught witnesses for the existential subformulae of φ : whenever a sort-1 existential has a witness in D for a parameter tuple drawn from the current stage, add one to D_0 ; whenever a sort-2 existential has a witness $S \in E$, add that S to E_0 . Iterating ω times yields D_0, E_0 that are *countable* (the language and the two ranges are countable, and each stage adds only countably many witnesses) and elementary for φ . By the Tarski–Vaught test, $\langle I, D_0, E_0 \rangle, g, G \models^d \psi \Leftrightarrow \langle I, D, E \rangle, g, G \models^d \psi$ for every subformula ψ of φ ; in particular $\langle I, D_0, E_0 \rangle, g, G \not\models^d \varphi$. The set-witnesses are honest elements of E , so $E_0 \sqsubseteq E$: *no comprehension is used*.

Step 2 (reindexing). D_0 and E_0 are countable and nonempty (they contain the nonempty ranges $\text{Range } g, \text{Range } G$). Since $V = V_2 = \text{nat}$ are infinite and only finitely many variables occur free in φ , choose g' agreeing with g on the free first-order variables of φ with $\text{Range } g' = D_0$, and G' agreeing with G on the free second-order variables with $\text{Range } G' = E_0$ — fix the finitely many free slots, and let the cofinitely many remaining slots enumerate the countable target.

Step 3 (conclude). By Coincidence, $\langle I, D_0, E_0 \rangle, g', G' \not\models^d \varphi$. But $D_0 = \text{Range } g'$ and $E_0 = \text{Range } G'$, hence $\langle I, \text{Range } g', \text{Range } G' \rangle, g', G' \not\models^d \varphi$, i.e. $\not\models_r^d \varphi$, as required.

Remark (the standard reading, exactly). The hull above is not only the engine of the converse; packaged with its truth-preservation it is a self-contained downward Löwenheim–Skolem theorem `DownwardLöwenheimSkolem`

— every countable nonempty seed sub-pair extends to a countable *elementary* substructure $\langle I, N, M \rangle \subseteq_E \langle I, D, E \rangle$. Applied to the full model and combined with FaithfulMD and the surjective reindexing, it yields the strongest faithfulness theorem `Deep'_to_MinS`: $\models^{d'} \varphi$ iff $\models^m (\llbracket \varphi \rrbracket)$ holds for every minimal interpretation whose range model is an elementary substructure of the full model (locale `MinS_ES_Univ`). The general reading is thus captured by the minimal embedding over *all* interpretations, the standard reading over the *elementary* ones; the gap between them is exactly the comprehension instance below.

Remark (why the standard reading is excluded by Löwenheim–Skolem). The construction stays within the general class precisely because the sort-2 witnesses are taken from the given E . It cannot reach the standard reading $\models^{d'} (E = \mathcal{P}(D))$ by an *all-interpretations* argument: by `comprehension_atom` the formula $\exists X. \forall x. (X(x) \leftrightarrow P(x, x))$ is $\models^{d'}$ -valid, yet a general model whose E omits the diagonal $\{d : P(d, d)\}$ refutes it, so it is not $\models^d$ -valid. Thus $\models^d \not\subseteq \models^{d'}$, and no range-relative Löwenheim–Skolem argument can bridge the two; the elementary restriction of `Deep'_to_MinS` cannot be relaxed to all interpretations. A structured Isabelle development of the argument above is provided as `MSOinHOL_lowenheim_skolem.thy`: the coincidence and reindexing lemmas, the two-sorted hull `ls_hull` (with its countable Tarski–Vaught construction `skolem_hull` and the truth preservation `truth_pres` it yields), the assembled converse, the elementary-substructure relation $\subseteq_E$ with `DownwardLowenheimSkolem` and `Deep'_to_MinS`, and this non-reduction remark are written out in full and machine-checked under Isabelle2025-2.

B Rendered Isabelle/HOL sources

For completeness every theory file of the development is rendered here as a syntax-highlighted source figure (the five theories already presented in the body — the deep embedding, the maximal- and minimal-shallow embeddings, faithfulness, and the experiments — are not repeated here). Long theories are split into consecutive parts of at most eighty lines; the line-number gutter is continuous, so the parts read as one listing. The full session (file `ROOT`) loads all theories in dependency order.

```

1 theory MSOinHOL_preliminaries imports Main
2 begin
3   <Some global settings.> declare[[syntax_ambiguity_warning=false]] nitpick_params[user_axioms,expect=genuine]
4   <Declarations shared between the deep, maximal-shallow, and minimal-shallow embeddings of MSO in HOL.>
5   typedecl D      <Domain of individuals.>
6   type_synonym R = nat      <Binary relation symbols (encoded as naturals).>
7   type_synonym V = nat      <First-order variable symbols (encoded as naturals).>
8   type_synonym V2 = nat     <Second-order (monadic) variable symbols (encoded as naturals).>
9   type_synonym R' = "D⇒D⇒bool" <Binary relations on individuals.>
10  type_synonym I = "R⇒R"    <Interpretations of relation symbols.>
11  type_synonym E = "V⇒D"    <First-order variable assignments.>
12  type_synonym G = "V2⇒(D⇒bool)" <Second-order variable assignments.>
13  type_synonym D' = "D⇒bool" <First-order domain restrictions (subsets of D as predicates).>
14  type_synonym P = "(D⇒bool)⇒bool" <Second-order domain restrictions (collections of sets as predicates).>
15  <Pointwise update of first-order variable assignments.>
16  definition EnvMod::"E⇒V⇒D⇒E" ("[_ ← _]" [110,0,0] 110) where "g[x←d] ≡ λz. if z = x then d else g z"
17  <Pointwise update of second-order variable assignments (distinct notation, angle brackets).>
18  definition SetMod::"G⇒V2⇒(D⇒bool)⇒G" ("⟦_ ← _⟧" [110,0,0] 110) where "G⟦X←S⟧ ≡ λZ. if Z = X then S else G Z"
19  <Standard lemmas about first-order variable-assignment update.>
20  lemma L1 [simp]: "x ≠ y ⇒ (g[y←d]) x = g x" by (simp add: EnvMod_def)
21  lemma L2: "a ≠ c ⇒ g[a←d₁][c←d₂] = g[c←d₂][a←d₁]" by (auto simp: EnvMod_def)
22  lemma L3 [simp]: "(g[a←d]) a = d" by (simp add: EnvMod_def)
23  lemma L4 [simp]: "g[a←d₁][a←d₂] = g[a←d₂]" by (auto simp: EnvMod_def)
24  <Standard lemmas about second-order variable-assignment update.>
25  lemma M1 [simp]: "X ≠ Y ⇒ (G⟦Y←S⟧) X = G X" by (simp add: SetMod_def)
26  lemma M2: "A ≠ C ⇒ G⟦A←S₁⟧⟦C←S₂⟧ = G⟦C←S₂⟧⟦A←S₁⟧" by (auto simp: SetMod_def)
27  lemma M3 [simp]: "(G⟦A←S⟧) A = S" by (simp add: SetMod_def)
28  lemma M4 [simp]: "G⟦A←S₁⟧⟦A←S₂⟧ = G⟦A←S₂⟧" by (auto simp: SetMod_def)
29  <Bounded quantifiers: <∀x:D. φ> stands for <∀x. D x ⇒ φ x> and <∃x:D. φ> stands for <∃x. D x ∧ φ x>.>
30  abbreviation "BAll D φ" ≡ ∀x. D x ⇒ φ x" syntax "BAll::"pttrn⇒logic⇒bool⇒bool" ("(3∀(_:_)/_)" [0,0,10]10)
31  translations "∀x:D. φ" ⇔ "CONST BAll D (λx. φ)"
32  abbreviation "BEx D φ" ≡ ∃x. D x ∧ φ x" syntax "BEx::"pttrn⇒logic⇒bool⇒bool" ("(3∃(_:_)/_)" [0,0,10]10)
33  translations "∃x:D. φ" ⇔ "CONST BEx D (λx. φ)"
34  <Set-as-predicate operations, range, and the universal domain (polymorphic; used for both sorts).>
35  abbreviation "Into" (infix "into" 100) where "f into D ≡ ∀x. D (f x)"
36  abbreviation "Range" where "Range f ≡ λx. ∃y. x = f y"
37  abbreviation "Onto" (infix "onto" 100) where "f onto D ≡ D = Range f"
38  abbreviation "Subset" (infix "⊆" 100) where "A ⊆ B ≡ ∀x. A x ⇒ B x"
39  abbreviation "Union" (infixl "∪" 110) where "A ∪ B ≡ λx. A x ∨ B x"
40  abbreviation "Univ" where "Univ ≡ λx. True"
41  <Surjectivity onto the universal predicate coincides with HOL surjectivity.>
42  lemma onto_Univ: "surj f = (f onto Univ)" by (auto simp: surj_def fun_eq_iff)
43  <Backward implication; useful for stating equivalences in two directions.>
44  abbreviation Bimp (infixr "⊆" 50) where "φ ⊆ ψ ≡ ψ ⇒ φ"
45 end

```

Fig. 6. Theory MSOinHOL_preliminaries.thy: the shared type declarations, the two variable-assignment update operators $g[x \leftarrow d]$ and $G\langle X \leftarrow S \rangle$ with their companion lemmata, the bounded quantifiers, and the polymorphic set-as-predicate operations.

```

1 theory MSOinHOL_shallow_minimal imports MSOinHOL_faithfulness_locale
2 begin
3   <Constants-only presentation: a global interpretation of the minimal shallow embedding.>
4   consts II::T gg::E GG::G
5   global_interpretation MinS II gg GG .
6   <Re-introduce the locale-internal notation at the global level.>
7   notation AtmM ("_m'(_)'") and PrdM ("_m'(_)'") and NegM ("¬_m" [58]59)
8   and AndM (infixr "∧^m" 56) and ExM (binder "∃^m" 53) and ExM2 (binder "∃^m_2" 53)
9   and OrM (infixr "∨^m" 54) and ImpM (infixr "⊃^m" 55) and IffM (infixr "⟷^m" 54)
10  and AllM (binder "∀^m" 53) and AllM2 (binder "∀^m_2" 53) and ValM ("⊨_m" 9) and DpToShM ("⊨")
11 end
    
```

Fig. 7. Theory `MSOinHOL_shallow_minimal.thy`: the constants-only global interpretation of `MinS` that re-issues the locale notation at the level of three uninterpreted constants `II,gg,GG`.

```

1 theory MSOinHOL_faithfulness imports MSOinHOL_shallow_minimal
2 begin
3   <Re-issuing the locale faithfulness theorems at the constants level.>
4   <Deep ↔ maximal shallow.>
5   theorem "(⟦(I,D,E),g,G ⊨^s ϕ⟧) ↔ (⟦(I,D,E),g,G ⊨^d ϕ⟧)" using FaithfulSDlem .
6   theorem "(⊨^s ϕ) ↔ (⊨^d ϕ)" using FaithfulSD .
7   <Deep ↔ minimal shallow, relative to the ranges of the chosen assignments gg and GG.>
8   theorem "(ϕ) ↔ ((I,Range gg,Range GG),gg,GG ⊨^d ϕ)" using FaithfulMDlem .
9   theorem "(⊨^m ϕ) ↔ ((I,Range gg,Range GG),gg,GG ⊨^d ϕ)" using FaithfulMD .
10  theorem "(ϕ) ↔ ((I,Range gg,Range GG),gg,GG ⊨^s ϕ)" using FaithfulMSlem .
11  theorem "(⊨^m ϕ) ↔ ((I,Range gg,Range GG),gg,GG ⊨^s ϕ)" using FaithfulMS .
12  <Global form across all interpretations and the one-directional bridge to full deep validity.>
13  theorem "(∀II gg GG. (⊨^m (MinS.DpToShM II gg GG ϕ))) = (∀I g G. (I,Range g,Range G),g,G ⊨^d ϕ)"
14  using FaithfulMS_all by (simp add: MinS.ValM_def)
15  theorem "(⊨^d ϕ) ⇒ (∀II gg GG. (⊨^m (MinS.DpToShM II gg GG ϕ)))"
16  using Deep_to_MinS by (simp add: MinS.ValM_def)
17  <Consistency check.>
18  lemma True nitpick[satisfy] oops
19 end
    
```

Fig. 8. Theory `MSOinHOL_faithfulness.thy`: the locale faithfulness theorems re-issued at the constants level (deep↔maximal- and deep↔minimal-shallow correspondences, the all-interpretations theorem, and the one-directional bridge to deep validity).

```

1 theory MS0inHOL_deep_subst_lemma imports MS0inHOL_deep
2 begin
3 —<Part A: first-order variable machinery.>
4 —<Free and bound first-order variable occurrences.>
5 primrec is_free (infix "free_in" 900) where
6   "x free_in (rd(u,v)) = (x = u ∨ x = v)"
7   | "x free_in (Xd(z)) = (x = z)"
8   | "x free_in (¬dφ) = (x free_in φ)"
9   | "x free_in (φ ∧d ψ) = (x free_in φ ∨ x free_in ψ)"
10  | "x free_in (∃dy. φ) = (x free_in φ ∧ x ≠ y)"
11  | "x free_in (∃dzY. φ) = (x free_in φ)"
12 abbreviation is_not_free (infix "not_free_in" 900) where "x not_free_in φ ≡ ¬(x free_in φ)"
13 fun is_bound (infix "bound_in" 900) where
14   "x bound_in (rd(u,v)) = False"
15   | "x bound_in (Xd(z)) = False"
16   | "x bound_in (¬dφ) = (x bound_in φ)"
17   | "x bound_in (φ ∧d ψ) = (x bound_in φ ∨ x bound_in ψ)"
18   | "x bound_in (∃dy. φ) = (x = y ∨ x bound_in φ)"
19   | "x bound_in (∃dzY. φ) = (x bound_in φ)"
20 abbreviation is_not_bound (infix "not_bound_in" 900) where "x not_bound_in φ ≡ ¬(x bound_in φ)"
21 abbreviation occurs (infix "occurs_in" 900) where "x occurs_in φ ≡ x free_in φ ∨ x bound_in φ"
22 abbreviation not_in (infix "not_in" 900) where "x not_in φ ≡ x not_free_in φ ∧ x not_bound_in φ"
23 —<A fresh first-order variable: strictly larger than every first-order variable occurring in φ.>
24 primrec fresh ("fresh(')") where
25   "fresh (rd(u,v)) = max (u+1) (v+1)"
26   | "fresh (Xd(z)) = z+1"
27   | "fresh (¬dφ) = fresh φ"
28   | "fresh (φ ∧d ψ) = max (fresh φ) (fresh ψ)"
29   | "fresh (∃dx. φ) = max (x+1) (fresh φ)"
30   | "fresh (∃dzY. φ) = fresh φ"
31 lemma L5: "x bound_in φ ⇒ x < (fresh φ)" by (induct φ) auto
32 lemma L6: "x free_in φ ⇒ x < (fresh φ)" by (induct φ) auto
33 lemma L7: "(fresh φ) not_in φ" using L5 L6 by blast
34 lemma L8: "max (fresh φ) (fresh ψ) not_free_in φ" by (metis L6 L7 max.absorb3 max_def)
35 lemma L9: "max (fresh φ) (fresh ψ) not_bound_in φ" by (metis L5 L7 max.absorb3 max_def)
36 lemma L10: "max (fresh φ) (fresh ψ) not_free_in ψ" by (metis L8 max.commute)
37 lemma L11: "max (fresh φ) (fresh ψ) not_bound_in ψ" by (metis L9 max.commute)
38 —<Irrelevance lemma: updating a non-free first-order variable does not affect truth.>
39 lemma L12: "y not_free_in φ ⇒ ((λD.E.g[y←d].G) ⊨d φ) = ((λD.E.g.G) ⊨d φ)"
40 by (induct φ arbitrary: G; simp; metis L4 L2)
41 —<First-order variable-for-variable substitution. The second-order binder descends transparently.>
42 primrec Subst ("["←"]") where
43   "[x←z](rd(u,v)) = rd((if x = u then z else u), (if x = v then z else v))"
44   | "[x←z](Xd(w)) = Xd((if x = w then z else w))"
45   | "[x←z](¬dφ) = ¬d([x←z](φ))"
46   | "[x←z](φ ∧d ψ) = ([x←z](φ) ∧d [x←z](ψ))"
47   | "[x←z](∃dy. φ) = (if x = y then (∃dy. φ) else (∃dy. [x←z](φ)))"
48   | "[x←z](∃dzY. φ) = (∃dzY. [x←z](φ))"
49 lemma L13 [simp]: "size [x←z](φ) = size φ" by (induct φ) auto
50 lemma L14 [simp]: "[x←x](φ) = φ" by (induct φ) auto
51 lemma L15: assumes "x ≠ a" shows "[a←z]([a←x](φ)) = [a←x](φ)" using assms by (induct φ) auto
52 lemma L16 [simp]: assumes "a ≠ x" shows "a not_free_in ([a←x](φ))" using assms by (induct φ) auto
53 —<Size-based induction principle (size-based on both existential binders).>
54 lemma SInduct:
55   assumes "⋀r u v. P (rd(u,v))"
56   and "⋀X z. P (Xd(z))"
57   and "⋀φ. (⋀ψ. size ψ ≤ size φ ⇒ P ψ) ⇒ P (¬dφ)"
58   and "⋀φ ψ. (⋀χ. size χ ≤ size φ + size ψ ⇒ P χ) ⇒ P (φ ∧d ψ)"
59   and "⋀y φ. (⋀ψ. size ψ ≤ size φ ⇒ P ψ) ⇒ P (∃dy. φ)"
60   and "⋀Y φ. (⋀ψ. size ψ ≤ size φ ⇒ P ψ) ⇒ P (∃dzY. φ)"
61   shows "P φ"
62 using assms proof (induct "size φ" arbitrary: φ rule: less_induct) case less thus ?case by (induct φ) auto qed
63 —<Stronger induction: structural for the propositional cases, size-based for the two binders.>
64 lemma QInduct:
65   assumes "⋀r u v. P (rd(u,v))"
66   and "⋀X z. P (Xd(z))"
67   and "⋀φ. P φ ⇒ P (¬dφ)"
68   and "⋀φ ψ. P φ ⇒ P ψ ⇒ P (φ ∧d ψ)"
69   and "⋀y φ. (⋀ψ. size ψ ≤ size φ ⇒ P ψ) ⇒ P (∃dy. φ)"
70   and "⋀Y φ. (⋀ψ. size ψ ≤ size φ ⇒ P ψ) ⇒ P (∃dzY. φ)"
71   shows "P φ"
72 using assms by (induct φ rule: SInduct) auto

```

Fig. 9. Theory MS0inHOL_deep_subst_lemma.thy (Part 1 of 3): Part A (first-order variables) — `is_free`, `is_bound`, `fresh` with their consequence lemmata, the irrelevance lemma L12, variable-for-variable substitution, and the size-based induction principles `SInduct`/`QInduct`.

Fig. 10. Theory `MS0inHOL_deep_subst_lemma.thy` (Part 2 of 3): the substitutability predicate, `SubstitutionLemma`, alphabetic renaming `ren_for_subst`, safe substitution `ren_subst`, and L17–L29 (Part A concluded); then the start of Part B (second-order variables) — `is_free2`, `is_bound2`, `fresh2`, and N5–N11.

```

146 —(Irrelevance lemma for second-order assignments.)
147 lemma N12: "Y not_free2_in  $\varphi \implies ((\langle I, D, E \rangle, g, G(Y \leftarrow S)) \models^d \varphi) = ((\langle I, D, E \rangle, g, G) \models^d \varphi)"$ 
148   by (induct  $\varphi$  arbitrary; g G; simp; metis M4 M2)
149 —(Second-order variable-for-variable substitution. The first-order binder descends transparently.)
150 primrec Subst2 ("[_  $\leftarrow$  _]('_)") where
151   "[X  $\leftarrow$  zZ](rd(u,v)) = rd(u,v)"
152   | "[X  $\leftarrow$  zZ](Yd(w)) = (if X = Y then Z else Y)d(w)"
153   | "[X  $\leftarrow$  zZ]( $\neg$ d $\varphi$ ) =  $\neg$ d([X  $\leftarrow$  zZ]( $\varphi$ ))"
154   | "[X  $\leftarrow$  zZ]( $\varphi \wedge$ d  $\psi$ ) = ([X  $\leftarrow$  zZ]( $\varphi$ )  $\wedge$ d [X  $\leftarrow$  zZ]( $\psi$ ))"
155   | "[X  $\leftarrow$  zZ]( $\exists$ dy.  $\varphi$ ) = ( $\exists$ dy. [X  $\leftarrow$  zZ]( $\varphi$ ))"
156   | "[X  $\leftarrow$  zZ]( $\exists$ dY.  $\varphi$ ) = (if X = Y then ( $\exists$ dY.  $\varphi$ ) else ( $\exists$ dY. [X  $\leftarrow$  zZ]( $\varphi$ )))"
157 lemma N13 [simp]: "size [X  $\leftarrow$  zZ]( $\varphi$ ) = size  $\varphi$ "
158 lemma N14 [simp]: "[X  $\leftarrow$  zX]( $\varphi$ ) =  $\varphi$ "
159 lemma N16 [simp]: assumes "A  $\neq$  X" shows "A not_free2_in ([A  $\leftarrow$  zX]( $\varphi$ ))"
160 primrec SubstitutableForIn2 ("[_ is'_subst2'_for _ in _] [999,1,999] 999) where
161   "Z is_subst2_for X in (rd(u,v)) = True"
162   | "Z is_subst2_for X in (Yd(w)) = True"
163   | "Z is_subst2_for X in ( $\neg$ d $\varphi$ ) = (Z is_subst2_for X in  $\varphi$ )"
164   | "Z is_subst2_for X in ( $\varphi \wedge$ d  $\psi$ ) = (Z is_subst2_for X in  $\varphi \wedge$  Z is_subst2_for X in  $\psi$ )"
165   | "Z is_subst2_for X in ( $\exists$ dy.  $\varphi$ ) = (Z is_subst2_for X in  $\varphi$ )"
166   | "Z is_subst2_for X in ( $\exists$ dY.  $\varphi$ ) = (Y = X  $\vee$  (X not_free2_in  $\varphi \vee Y \neq Z \wedge$  Z is_subst2_for X in  $\varphi$ ))"
167 lemma SubstitutionLemma2 [simp]:
168   assumes "Z is_subst2_for X in  $\varphi$ " shows "((\langle I, D, E \rangle, g, G)  $\models^d$  ([X  $\leftarrow$  zZ]( $\varphi$ ))) = ((\langle I, D, E \rangle, g, G(X  $\leftarrow$  (G Z)))  $\models^d$   $\varphi$ )"
169   using assms by (induction  $\varphi$  arbitrary; g G; auto simp: N12 M2)
170 fun ren_for_subst2 where
171   "ren_for_subst2 X Z (rd(u,v)) = rd(u,v)"
172   | "ren_for_subst2 X Z (Yd(w)) = Yd(w)"
173   | "ren_for_subst2 X Z ( $\neg$ d $\varphi$ ) =  $\neg$ d(ren_for_subst2 X Z  $\varphi$ )"
174   | "ren_for_subst2 X Z ( $\varphi \wedge$ d  $\psi$ ) = (ren_for_subst2 X Z  $\varphi \wedge$ d ren_for_subst2 X Z  $\psi$ )"
175   | "ren_for_subst2 X Z ( $\exists$ dy.  $\varphi$ ) =  $\exists$ dy. ren_for_subst2 X Z  $\varphi$ "
176   | "ren_for_subst2 X Z ( $\exists$ dY.  $\varphi$ ) = (if Y = Z  $\wedge$  X free2_in  $\varphi$ 
177     then let f = max (fresh2  $\varphi$ ) (Z+1);  $\varphi'$  = [Y  $\leftarrow$  zf]( $\varphi$ ) in ( $\exists$ dY. ren_for_subst2 X Z  $\varphi'$ )
178     else  $\exists$ dY. ren_for_subst2 X Z  $\varphi$ )"
179 lemma N17 [simp]: "size (ren_for_subst2 X Z  $\varphi$ ) = size  $\varphi$ "
180 lemma N18: " $\alpha$  not2_in  $\varphi \implies \alpha$  is_subst2_for  $\beta$  in  $\varphi$ "
181 lemma N19: "X free2_in  $\psi \implies Y \neq X \implies X$  free2_in [Y  $\leftarrow$  zZ]( $\psi$ )"
182 lemma N20 [simp]: "X free2_in ren_for_subst2 X Z  $\varphi$  = (X free2_in  $\varphi$ )"
183   by (induct  $\varphi$  rule: QInduct; simp add: Let_def) (metis N16 N6 N7 N19 max.absorb3 max_def_raw)
184 lemma N21: "((\langle I, D, E \rangle, g, G)  $\models^d$   $\varphi$ ) = ((\langle I, D, E \rangle, g, G)  $\models^d$  (ren_for_subst2 X Z  $\varphi$ ))"
185   rule: QInduct; simp add: Let_def; smt (verit) N12 N18 M2 M3 N5 N6 SubstitutionLemma2 max.strict_order_iff max_def
186 lemma N22: "Z is_subst2_for X in (ren_for_subst2 X Z  $\varphi$ )"
187   metis N13 SubstitutableForIn2.simps(6) Suc_n_not_le_n_dual_order.refl max.cobounded2
188 definition ren_subst2 ("[_  $\leftarrow$  _]('_)") where "[X  $\leftarrow$  zZ]( $\varphi$ ) = [X  $\leftarrow$  zZ](ren_for_subst2 X Z  $\varphi$ )"
189 lemma N27 [simp]: "((\langle I, D, E \rangle, g, G)  $\models^d$  [X  $\leftarrow$  zZ]( $\varphi$ )) = ((\langle I, D, E \rangle, g, G(X  $\leftarrow$  (G Z)))  $\models^d$   $\varphi$ )"
190 lemma N28 [simp]: "size ([X  $\leftarrow$  zZ]( $\varphi$ )) = size  $\varphi$ "
191 lemma N29: "G onto E  $\implies$  ((\langle I, D, E \rangle, g, G)  $\models^d$  ( $\exists$ dX.  $\varphi$ )) = ( $\exists$ Z. (\langle I, D, E \rangle, g, G)  $\models^d$  ([X  $\leftarrow$  zZ]( $\varphi$ )))"
192   by (induct  $\varphi$  arbitrary; I G G X rule: QInduct; simp; blast)
193 end

```

Fig. 11. Theory MS0inHOL.deep_subst_lemma.thy (Part 3 of 3): Part B (second-order variables) concluded — the irrelevance lemma N12, second-order substitution Subst2, SubstitutionLemma2, renaming ren_for_subst2/ren_subst2, and N17–N29.

```

1 theory MSOinHOL_subst_extras imports MSOinHOL_deep_subst_lemma
2 begin
3 <Explicit rename-evaluation lemmas: renaming a bound variable to a fresh f and updating the assignment
4 preserves truth -- the semantic core behind L21 / N21 (a future smt-free reproof).>
5 <First-order: rename y to a fresh f, then evaluate.>
6 lemma rename_eval:
7   assumes "fresh  $\varphi \leq f$ " and " $y < f$ " shows " $((I,D,E),g[f \leftarrow d],G \models^d [y \leftarrow f](\varphi)) = ((I,D,E),g[y \leftarrow d],G \models^d \varphi)$ "
8 proof -
9   have fy: " $f \neq y$ " using assms(2) by simp
10  have nf: " $f$  not_free_in  $\varphi$ " using assms(1) by (meson L6 leD)
11  have "f not_in  $\varphi$ " using assms(1) by (meson L5 L6 leD)
12  hence sub: " $f$  is_subst_for y in  $\varphi$ " by (rule L18)
13  have swap: " $(g[f \leftarrow d])[y \leftarrow d] = (g[y \leftarrow d])[f \leftarrow d]$ " using fy by (rule L2)
14  have " $((I,D,E),g[f \leftarrow d],G \models^d [y \leftarrow f](\varphi)) = ((I,D,E),g[f \leftarrow d],G \models^d ((g[f \leftarrow d]) f))$ "
15    using sub by (rule SubstitutionLemma)
16  also have "... =  $((I,D,E),g[f \leftarrow d],G \models^d \varphi)$ " by simp
17  also have "... =  $((I,D,E),g[y \leftarrow d],G \models^d \varphi)$ " using swap by simp
18  also have "... =  $((I,D,E),g[y \leftarrow d],G \models^d \varphi)$ " using nf by (simp add: L12)
19  finally show ?thesis .
20 qed
21 <Second-order: rename Y to a fresh f, then evaluate. (The monadic-set twin of rename_eval.)>
22 lemma rename_eval2:
23   assumes "fresh2  $\varphi \leq f$ " and " $Y < f$ " shows " $((I,D,E),g,G(f \leftarrow S) \models^d [Y \leftarrow zf](\varphi)) = ((I,D,E),g,G(Y \leftarrow S) \models^d \varphi)$ "
24 proof -
25   have fy: " $f \neq Y$ " using assms(2) by simp
26   have nf: " $f$  not_free2_in  $\varphi$ " using assms(1) by (meson N6 leD)
27   have "f not2_in  $\varphi$ " using assms(1) by (meson N5 N6 leD)
28   hence sub: " $f$  is_subst2_for Y in  $\varphi$ " by (rule N18)
29   have swap: " $G(f \leftarrow S)(Y \leftarrow S) = G(Y \leftarrow S)(f \leftarrow S)$ " using fy by (rule M2)
30   have " $((I,D,E),g,G(f \leftarrow S) \models^d [Y \leftarrow zf](\varphi)) = ((I,D,E),g,G(f \leftarrow S)(Y \leftarrow ((G(f \leftarrow S)) f)) \models^d \varphi)$ "
31     using sub by (rule SubstitutionLemma2)
32   also have "... =  $((I,D,E),g,G(f \leftarrow S)(Y \leftarrow S) \models^d \varphi)$ " by simp
33   also have "... =  $((I,D,E),g,G(Y \leftarrow S)(f \leftarrow S) \models^d \varphi)$ " using swap by simp
34   also have "... =  $((I,D,E),g,G(Y \leftarrow S) \models^d \varphi)$ " using nf by (simp add: N12)
35   finally show ?thesis .
36 qed
37 end

```

Fig. 12. Theory MSOinHOL_subst_extras.thy: the explicit rename-evaluation lemmas rename_eval (first-order) and rename_eval2 (second-order).

```

1 theory MSOinHOL_comprehension imports MSOinHOL_deep_subst_lemma
2 begin
3 —<Monadic comprehension as a derived schema: under the full second-order domain (@{text "ValD"}, E = Univ)
4 comprehension holds for every  $\varphi$  with X not free; the witness is the set defined by  $\varphi$ .>
5 theorem comprehension_schema: assumes "X not_free2_in  $\varphi$ " shows " $\models^d (\exists^d_2 X. \forall^d x. ((X^d(x)) \longleftrightarrow^d \varphi))$ "
6 unfolding ValD'_def
7 proof (intro allI)
8 fix I g G
9 let ?S = " $\lambda d. \langle I, \text{Univ}, \text{Univ} \rangle, g[x \leftarrow d], G \models^d \varphi$ "
10 have " $\langle I, \text{Univ}, \text{Univ} \rangle, g, G \langle X \leftarrow ?S \rangle \models^d (\forall^d x. ((X^d(x)) \longleftrightarrow^d \varphi))$ " using assms by (simp add: DefD N12)
11 thus " $\langle I, \text{Univ}, \text{Univ} \rangle, g, G \models^d (\exists^d_2 X. \forall^d x. ((X^d(x)) \longleftrightarrow^d \varphi))$ " by auto
12 qed
13 —<Headline instance (cf. comprehension_d): the set of r-self-related individuals exists.>
14 corollary comprehension_atom: " $\models^d (\exists^d_2 X. \forall^d x. ((X^d(x)) \longleftrightarrow^d (r^d(x,x))))$ " by (rule comprehension_schema) simp
15 —<Standard vs. weak (Henkin): comprehension_schema needs E = Univ and fails for general @{text "ValD"}
16 -- the standard-vs-Henkin gap behind the Loewenheim-Skolem problem (see the paper).>
17 end

```

Fig. 13. Theory `MSOinHOL_comprehension.thy`: the general monadic comprehension_schema over the full second-order domain with its witness, the corollary `comprehension_atom`, and the discussion relating comprehension to standard vs. general (Henkin-style) semantics.

```

1 theory MSOinHOL_experiments_classic imports MSOinHOL_deep MSOinHOL_shallow MSOinHOL_shallow_minimal
2 begin
3 abbreviation "(x::V) ≡ 1" abbreviation "(y::V) ≡ 2" abbreviation "(z::V) ≡ 3"
4 abbreviation "(X::V2) ≡ 1" abbreviation "(Y::V2) ≡ 2" abbreviation "(Z::V2) ≡ 3"
5 abbreviation "(u::V) ≡ 4" abbreviation "(v::V) ≡ 5" consts P::R
6 <Boolean closure (Büchi 1960; Thomas 1997) under @ {text "≡ d"}.>
7 lemma complement_d: "≡ d (∀ d2X. ∃ d2Z. ∀ dX. (Z d(x) ⟷ d ¬d X d(x)))"
8 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
9 subgoal for S by (rule exI[of _ "λd. ¬ S d"]) (auto simp: SetMod_def EnvMod_def) done
10 lemma intersection_d: "≡ d (∀ d2X. ∀ d2Y. ∃ d2Z. ∀ dX. (Z d(x) ⟷ d (X d(x) ∧ d Y d(x))))"
11 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
12 subgoal for S Sa by (rule exI[of _ "λd. S d ∧ Sa d"]) (auto simp: SetMod_def EnvMod_def) done
13 lemma union_d: "≡ d (∀ d2X. ∀ d2Y. ∃ d2Z. ∀ dX. (Z d(x) ⟷ d (X d(x) ∨ d Y d(x))))"
14 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
15 subgoal for S Sa by (rule exI[of _ "λd. S d ∨ Sa d"]) (auto simp: SetMod_def EnvMod_def) done
16 <Graph operations (Courcelle 2012).>
17 lemma separation_d: "≡ d (∀ d2X. ∃ d2Z. ∀ dX. (Z d(x) ⟷ d (X d(x) ∧ d P d(x,x))))"
18 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
19 subgoal for I S by (rule exI[of _ "λd. S d ∧ I P d d"]) (auto simp: SetMod_def EnvMod_def) done
20 lemma image_d: "≡ d (∀ d2X. ∃ d2Y. ∀ dX. (Y d(x) ⟷ d ∃ dY. (X d(y) ∧ d P d(y,x))))"
21 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
22 subgoal for I S by (rule exI[of _ "λd. ∃ d'. S d' ∧ I P d' d"]) (auto simp: SetMod_def EnvMod_def) done
23 lemma preimage_d: "≡ d (∀ d2X. ∃ d2Y. ∀ dX. (Y d(x) ⟷ d ∃ dY. (P d(x,y) ∧ d X d(y))))"
24 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
25 subgoal for I S by (rule exI[of _ "λd. ∃ d'. I P d' d ∧ S d"]) (auto simp: SetMod_def EnvMod_def) done
26 <Reachability (Basin & Klarlund 1995): not universally valid; reflexive variant is.>
27 lemma reachability_not_valid_d: "≡ d (∀ d2Z. ((Z d(x) ∧ d (∀ d. (Z d(u) ⊃ d ∀ dV. (P d(u,v) ⊃ d Z d(v)))) ⊃ d Z d(y)))"
28 unfolding DefD apply simp nitpick oops
29 lemma reachability_reflexive_d: "≡ d (∀ d2Z. ((Z d(x) ∧ d (∀ d. (Z d(u) ⊃ d ∀ dV. (P d(u,v) ⊃ d Z d(v)))) ⊃ d Z d(x)))"
30 unfolding DefD by simp
31 <2-colorability (Thomas 1997): refuted on SK_3S.>
32 lemma two_colorability_not_valid_d: "≡ d (∃ d2Z. ∀ dX. (P d(x,y) ⊃ d (Z d(x) ⟷ d ¬d Z d(y))))"
33 unfolding DefD apply simp nitpick oops
34 <Same landmarks in the maximal-shallow embedding: structurally identical proofs.>
35 lemma complement_s: "≡ s (∀ s2X. ∃ s2Z. ∀ sX. (Z s(x) ⟷ s ¬s X s(x)))"
36 unfolding DefS apply (intro allI, simp, intro allI) subgoal for S by (rule exI[of _ "λd. ¬ S d"]) auto done
37 lemma intersection_s: "≡ s (∀ s2X. ∀ s2Y. ∃ s2Z. ∀ sX. (Z s(x) ⟷ s (X s(x) ∧ s Y s(x))))"
38 unfolding DefS apply (intro allI, simp, intro allI) subgoal for S Sa by (rule exI[of _ "λd. S d ∧ Sa d"]) auto done
39 lemma union_s: "≡ s (∀ s2X. ∀ s2Y. ∃ s2Z. ∀ sX. (Z s(x) ⟷ s (X s(x) ∨ s Y s(x))))"
40 unfolding DefS apply (intro allI, simp, intro allI) subgoal for S Sa by (rule exI[of _ "λd. S d ∨ Sa d"]) auto done
41 lemma separation_s: "≡ s (∀ s2X. ∃ s2Z. ∀ sX. (Z s(x) ⟷ s (X s(x) ∧ s P s(x,x))))"
42 unfolding DefS apply (intro allI, simp, intro allI) subgoal for I S by (rule exI[of _ "λd. S d ∧ I P d d"]) auto done
43 lemma image_s: "≡ s (∀ s2X. ∃ s2Y. ∀ sX. (Y s(x) ⟷ s ∃ sY. (X s(y) ∧ s P s(y,x))))"
44 unfolding DefS apply (intro allI, simp, intro allI) subgoal for I S by (rule exI[of _ "λd. ∃ d'. S d' ∧ I P d' d"]) auto done
45 lemma preimage_s: "≡ s (∀ s2X. ∃ s2Y. ∀ sX. (Y s(x) ⟷ s ∃ sY. (P s(x,y) ∧ s X s(y))))"
46 unfolding DefS apply (intro allI, simp, intro allI) subgoal for I S by (rule exI[of _ "λd. ∃ d'. I P d' d ∧ S d"]) auto done
47 lemma reachability_not_valid_s: "≡ s (∀ s2Z. ((Z s(x) ∧ s (∀ d. (Z s(u) ⊃ s (∀ dV. (P s(u,v) ⊃ s Z s(v)))) ⊃ s Z s(y))))"
48 unfolding DefS apply (intro allI) apply simp nitpick oops
49 lemma reachability_reflexive_s: "≡ s (∀ s2Z. ((Z s(x) ∧ s (∀ d. (Z s(u) ⊃ s (∀ dV. (P s(u,v) ⊃ s Z s(v)))) ⊃ s Z s(x))))"
50 unfolding DefS by simp
51 lemma two_colorability_not_valid_s: "≡ s (∃ s2Z. ∀ sX. ∀ sY. (P s(x,y) ⊃ s (Z s(x) ⟷ s ¬s Z s(y))))"
52 unfolding DefS apply (intro allI) apply simp nitpick oops
53 <Minimal embedding: SO quantifier ranges over the countable Range GG (via nat), not all of Pow(D).>
54 Nitpick can only certify a POTENTIAL countermodel.>
55 lemma complement_m_not_valid: "≡ m (∀ m2X. ∃ m2Z. ∀ mX. (Z m(x) ⟷ m ¬m X m(x)))"
56 unfolding DefM nitpick[expect=potential] oops
57 lemma intersection_m_not_valid: "≡ m (∀ m2X. ∀ m2Y. ∃ m2Z. ∀ mX. (Z m(x) ⟷ m (X m(x) ∧ m Y m(x))))"
58 unfolding DefM nitpick[expect=potential] oops
59 lemma reachability_not_valid_m: "≡ m (∀ m2Z. ((Z m(x) ∧ m (∀ m. (Z m(u) ⊃ m (∀ mV. (P m(u,v) ⊃ m Z m(v)))) ⊃ m Z m(y))))"
60 unfolding DefM nitpick[expect=potential] oops
61 <Reflexive reachability: the conclusion is the first conjunct of the antecedent; genuinely valid.>
62 lemma reachability_reflexive_m: "≡ m (∀ m2Z. ((Z m(x) ∧ m (∀ m. (Z m(u) ⊃ m (∀ mV. (P m(u,v) ⊃ m Z m(v)))) ⊃ m Z m(x))))"
63 unfolding DefM by simp
64 lemma two_colorability_not_valid_m: "≡ m (∃ m2Z. ∀ mX. ∀ mY. (P m(x,y) ⊃ m (Z m(x) ⟷ m ¬m Z m(y))))"
65 unfolding DefM nitpick[expect=potential] oops
66 end

```

Fig. 14. Theory `MSOinHOL_experiments_classic.thy`: the classical MSO landmarks in the deep and maximal-shallow embeddings — Boolean closure (complement, intersection, union), monadic graph operations (separation, P -image, P -preimage), reachability, and 2-colorability — each with an explicit second-order witness via `exI`; and the minimal embedding, where Boolean closure and reachability fail by `nitpick` while reflexive reachability remains valid.

```

1 theory MS0inHOL_experiments_locale imports MS0inHOL_faithfulness_locale
2 begin
3   <Demonstration: minimal-shallow proofs that hold in every interpretation of the MinS locale transfer to validity in the
4     model class whose domains are the assignment ranges.>
5   <Derived rule: from validity in every minimal interpretation to range-relative deep validity.>
6   lemmas MinS_to_Deep = FaithfulMS_all[THEN iffD1, rule_format]
7   <Cosmetic locale-level rewrite rules for derived connectives under DpToShM.>
8   lemma (in MinS) OrM_simp[DefM]: " $\langle\varphi \vee^d \psi\rangle = \langle\varphi\rangle \vee^m \langle\psi\rangle$ " by (simp add: OrD_def OrM_def)
9   lemma (in MinS) ImpM_simp[DefM]: " $\langle\varphi \supset^d \psi\rangle = \langle\varphi\rangle \supset^m \langle\psi\rangle$ " by (simp add: ImpD_def DefM)
10  <Example: a membership tautology proved using only minimal-shallow infrastructure.>
11  lemma (in MinS) Mem_MinS: " $\models^m \langle(P^d(x,x)) \supset^d (P^d(x,x))\rangle$ " by (auto simp: DefM)
12  <Transfer to the range-relative deep embedding requires no further work.>
13  lemma " $\langle l, \text{Range } g, \text{Range } G \rangle, g, G \models^d \langle(P^d(x,x)) \supset^d (P^d(x,x))\rangle$ " apply (rule MinS_to_Deep) by (blast intro: MinS.Mem_MinS)
14 end

```

Fig. 15. Theory MS0inHOL_experiments_locale.thy: locale-based transfer to the deep embedding — the directional reading MinS_to_Deep of FaithfulMS_all, two cosmetic rewrite rules, a schema proved inside the locale, and its one-line transfer.

```

1 theory MSOinHOL_lowenheim_skolem_lemmas
2 imports MSOinHOL_faithfulness_locale MSOinHOL_comprehension "HOL-Library.Countable_Set"
3 begin
4   <Supporting lemmas for the downward Löwenheim-Skolem construction in MSOinHOL_lowenheim_skolem.
5   This file holds the technical machinery (Skolem-witness operators wF / wS, the omega-stage iteration
6   'stage', monotonicity / countability / common-stage lemmas). The main theorems live in the parent file.>
7   <coincidence (generalises L12 / N12): truth depends only on the free variables of  $\varphi$ . Compact structured induction;
8   the binder cases use IH after extending the agreement through the bound-variable update via EnvMod_def / SetMod_def.>
9 lemma coincidence:
10   assumes " $\wedge x. x \text{ free\_in } \varphi \implies g \ x = g' \ x$ " and " $\wedge X. X \text{ free2\_in } \varphi \implies G \ X = G' \ X$ "
11   shows " $((\langle I, D, E \rangle, g, G) \models^d \varphi) = ((\langle I, D, E \rangle, g', G') \models^d \varphi)$ "
12   using assms
13   proof (induct  $\varphi$  arbitrary: g g' G G')
14   case (AtmD r u v) thus ?case by simp next
15   case (PrdD X z) thus ?case by simp next
16   case (NegD  $\varphi$ ) have " $((\langle I, D, E \rangle, g, G) \models^d \varphi) = ((\langle I, D, E \rangle, g', G') \models^d \varphi)$ "
17   using NegD.hyps NegD.prem1,2 is_free.simps(3) is_free2.simps(3) by presburger thus ?case by simp next
18   case (AndD  $\varphi \ \psi$ ) from AndD.hyps[of g g' G G'] AndD.prem1 show ?case by simp next
19   case (ExD y  $\varphi$ ) have " $((\langle I, D, E \rangle, g[y \leftarrow d], G) \models^d \varphi) = ((\langle I, D, E \rangle, g'[y \leftarrow d], G') \models^d \varphi)$ " for d
20   using EnvMod_def ExD.hyps ExD.prem1,2 is_free.simps(5) is_free2.simps(5) by presburger thus ?case by simp next
21   case (ExD2 Y  $\varphi$ ) have " $((\langle I, D, E \rangle, g, G(Y \leftarrow S)) \models^d \varphi) = ((\langle I, D, E \rangle, g', G'(Y \leftarrow S)) \models^d \varphi)$ " for S
22   by (rule ExD2.hyps) (use ExD2.prem1 in <auto simp: SetMod_def>) thus ?case by simp
23 qed
24 <The formula type  $\mathcal{F}$  is countable (datatype over countable atoms).>
25 instance  $\mathcal{F} :: \text{countable}$  by countable_datatype
26 <Pad a finite parameter list to a total assignment (default outside its range).>
27 definition padA :: " $D \Rightarrow D \text{ list} \Rightarrow (V \Rightarrow D)$ " where "padA d0 xs = ( $\lambda i. \text{if } i < \text{length } xs \text{ then } xs ! i \text{ else } d0$ )"
28 definition padB :: " $(D \Rightarrow \text{bool}) \Rightarrow (D \Rightarrow \text{bool}) \text{ list} \Rightarrow (V2 \Rightarrow (D \Rightarrow \text{bool}))$ " where
29   "padB S0 Ss = ( $\lambda i. \text{if } i < \text{length } Ss \text{ then } Ss ! i \text{ else } S0$ )"
30 <FO / SO Skolem witnesses for one demand, drawn from the big model  $(\langle I, D, E \rangle)$ .>
31 definition wF :: " $\mathcal{I} \Rightarrow (D \Rightarrow \text{bool}) \Rightarrow ((D \Rightarrow \text{bool}) \Rightarrow \text{bool}) \Rightarrow D \Rightarrow (D \Rightarrow \text{bool}) \Rightarrow \mathcal{F} \Rightarrow V \Rightarrow D \text{ list} \Rightarrow (D \Rightarrow \text{bool}) \text{ list} \Rightarrow D$ " where
32   "wF I D E d0 S0  $\psi$  y xs Ss = (if ( $\exists d. D \ d \wedge (\langle I, D, E \rangle, (\text{padA } d0 \ xs)[y \leftarrow d], (\text{padB } S0 \ Ss)) \models^d \psi$ )
33     then (SOME d.  $D \ d \wedge (\langle I, D, E \rangle, (\text{padA } d0 \ xs)[y \leftarrow d], (\text{padB } S0 \ Ss)) \models^d \psi$ )
34     else d0)"
35 definition wS :: " $\mathcal{I} \Rightarrow (D \Rightarrow \text{bool}) \Rightarrow ((D \Rightarrow \text{bool}) \Rightarrow \text{bool}) \Rightarrow D \Rightarrow (D \Rightarrow \text{bool}) \Rightarrow \mathcal{F} \Rightarrow V2 \Rightarrow D \text{ list} \Rightarrow (D \Rightarrow \text{bool}) \text{ list} \Rightarrow (D \Rightarrow \text{bool})$ " where
36   "wS I D E d0 S0  $\psi$  Y xs Ss = (if ( $\exists S. E \ S \wedge (\langle I, D, E \rangle, (\text{padA } d0 \ xs), (\text{padB } S0 \ Ss)(Y \leftarrow S)) \models^d \psi$ )
37     then (SOME S.  $E \ S \wedge (\langle I, D, E \rangle, (\text{padA } d0 \ xs), (\text{padB } S0 \ Ss)(Y \leftarrow S)) \models^d \psi$ )
38     else S0)"
39 lemma wF_mem: " $D \ d0 \implies D \ (wF \ I \ D \ E \ d0 \ S0 \ \psi \ y \ xs \ Ss)$ " unfolding wF_def
40   by (cases "d0" d d'  $\wedge (\langle I, D, E \rangle, (\text{padA } d0 \ xs)[y \leftarrow d], (\text{padB } S0 \ Ss)) \models^d \psi$ ; simp; metis (mono_tags, lifting) someI_ex)
41 lemma wS_mem: " $E \ S0 \implies E \ (wS \ I \ D \ E \ d0 \ S0 \ \psi \ Y \ xs \ Ss)$ " unfolding wS_def
42   by (cases "S0" S S'  $\wedge (\langle I, D, E \rangle, (\text{padA } d0 \ xs), (\text{padB } S0 \ Ss)(Y \leftarrow S)) \models^d \psi$ ; simp; metis (mono_tags, lifting) someI_ex)
43 <Witness-membership specialised to seeds @ {text "g 0"} / @ {text "G 0"}: no extra "D (g 0)" premise.>
44 lemma wF_mem_seed: "g into D  $\implies D \ (wF \ I \ D \ E \ (g \ 0) \ S0 \ \psi \ y \ xs \ Ss)$ " by (blast intro: wF_mem)
45 lemma wS_mem_seed: "G into E  $\implies E \ (wS \ I \ D \ E \ (G \ 0) \ \psi \ Y \ xs \ Ss)$ " by (blast intro: wS_mem)
46 <The omega-stage (FQ carrier, SO carrier): step n+1 adds every wF / wS witness over parameter lists drawn from
47 the carriers at stage n.>
48 primrec stage :: " $\mathcal{I} \Rightarrow (D \Rightarrow \text{bool}) \Rightarrow ((D \Rightarrow \text{bool}) \Rightarrow \text{bool}) \Rightarrow (V \Rightarrow D) \Rightarrow (V2 \Rightarrow (D \Rightarrow \text{bool})) \Rightarrow \text{nat} \Rightarrow ((D \Rightarrow \text{bool}) \times ((D \Rightarrow \text{bool}) \Rightarrow \text{bool}))$ " where
49   "stage I D E g G 0 = (Range g, Range G)"
50 | "stage I D E g G (Suc n) =
51   (( $\lambda d. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ d$ 
52      $\vee (\exists \psi \ y \ xs \ Ss. (\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ x) \wedge (\forall S \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S) \wedge d = wF \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ y \ xs \ Ss)$ ,
53     ( $\lambda S. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S$ 
54        $\vee (\exists \psi \ Y \ xs \ Ss. (\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ x) \wedge (\forall S' \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S') \wedge S = wS \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ Y \ xs \ Ss))$ )"
55 <Helper: countability of a binary product (from countable_SIGMA).>
56 lemma countable_Times2: "countable A  $\implies$  countable B  $\implies$  countable (A  $\times$  B)"
57 using countable_SIGMA[of A "A" B] by (simp add: Sigma_def)
58 <An explicit Skolem witness produced from parameters drawn from stage n sits in stage (Suc n) (FO / SO).>
59 Direct from the second clause of 'stage'.
60 lemma fst_stage_Suc_intro:
61   " $\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ x \implies \forall S \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S$ 
62    $\implies \text{fst } (\text{stage } I \ D \ E \ g \ G \ (\text{Suc } n)) \ (wF \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ y \ xs \ Ss)$ " by (simp, blast)
63 lemma snd_stage_Suc_intro:
64   " $\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ x \implies \forall S \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S$ 
65    $\implies \text{snd } (\text{stage } I \ D \ E \ g \ G \ (\text{Suc } n)) \ (wS \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ Y \ xs \ Ss)$ " by (simp, blast)
66 <coincidence applied to padded assignments built from a/b on the free-variable prefix of  $\psi$ :
67 the padded truth equals the original. Drives the TV-step truth transfer automatically.>
68 lemma pad_truth_eq_FO:
69   " $((\langle I, D, E \rangle, (\text{padA } d0 \ (\text{map } a \ [0..<\text{fresh } \psi])][y \leftarrow d], (\text{padB } S0 \ (\text{map } b \ [0..<\text{fresh2 } \psi])) \models^d \psi) = ((\langle I, D, E \rangle, a[y \leftarrow d], b) \models^d \psi)$ "
70   by (rule coincidence) (auto dest!: L6 N6 simp: EnvMod_def padA_def padB_def)
71 lemma pad_truth_eq_SO:
72   " $((\langle I, D, E \rangle, \text{padA } d0 \ (\text{map } a \ [0..<\text{fresh } \psi]), (\text{padB } S0 \ (\text{map } b \ [0..<\text{fresh2 } \psi]))(Y \leftarrow S)) \models^d \psi = ((\langle I, D, E \rangle, a, b(Y \leftarrow S)) \models^d \psi)$ "
73   by (rule coincidence) (auto dest!: L6 N6 simp: SetMod_def padA_def padB_def)

```

Fig. 16. Theory MSOinHOL_lowenheim_skolem_lemmas.thy (Part 1 of 4): the supporting machinery for the Löwenheim–Skolem construction — the coincidence lemma coincidence (a two-sorted generalisation of L12/N12), the padding maps padA/padB, the Skolem-witness operators wF/wS with their membership lemmas, the ω -stage operator stage, the stage-introduction lemmas fst_stage_Suc_intro/snd_stage_Suc_intro, and the padding-truth lemmas pad_truth_eq_FO/pad_truth_eq_SO.

```

76 --<The Skolem witness wF / wS realizes its demand whenever one exists in (D,E): membership in D / E plus
77 the truth on the padded assignment. Direct from 'some_ex'.>
78 lemma wF_realizes:
79   assumes "∃d. D d ∧ (I,D,E).(padA d0 xs)[y←d].(padB S0 Ss) ⊨d ψ"
80   shows "D (wF I D E d0 S0 ψ y xs Ss) ∧ ((I,D,E).(padA d0 xs)[y←wF I D E d0 S0 ψ y xs Ss].(padB S0 Ss) ⊨d ψ)"
81   using some_ex[OF assms] assms unfolding wF_def by simp
82 lemma wS_realizes:
83   assumes "∃S. E S ∧ (I,D,E).(padA d0 xs).(padB S0 Ss)(Y←S) ⊨d ψ"
84   shows "E (wS I D E d0 S0 ψ Y xs Ss) ∧ ((I,D,E).(padA d0 xs).(padB S0 Ss)(Y←wS I D E d0 S0 ψ Y xs Ss) ⊨d ψ)"
85   using some_ex[OF assms] assms unfolding wS_def by simp
86 --<Full Tarski-Vaught step (FO / SO): from an existential over (D,E) reachable through a, b, the wF / wS witness
87 lives in D / E and satisfies the formula relative to a, b. Core lemma for the TVfo / TVso of 'skolem_hull'.>
88 lemma skolem_step_FO:
89   assumes "∃d. D d ∧ (I,D,E).a[y←d].b ⊨d ψ"
90   shows "D (wF I D E d0 S0 ψ y (map a [0..d ψ"
92 proof -
93   let ?xs = "map a [0..d ψ" using assms by (simp add: pad_truth_eq_FO)
95   from wF_realizes[OF this] show ?thesis by (simp add: pad_truth_eq_FO)
96 qed
97 lemma skolem_step_SO:
98   assumes "∃S. E S ∧ (I,D,E).a.b(Y←S) ⊨d ψ"
99   shows "E (wS I D E d0 S0 ψ Y (map a [0..d ψ"
101 proof -
102   let ?xs = "map a [0..d ψ" using assms by (simp add: pad_truth_eq_SO)
104   from wS_realizes[OF this] show ?thesis by (simp add: pad_truth_eq_SO)
105 qed
106 --<The (Suc n)-stage is contained in the n-stage plus the image of wF / wS over UNIV × UNIV × (lists from n).
107 Used to derive countability of each stage by induction.>
108 lemma fst_stage_Suc_subset:
109   "{d. fst (stage I D E g G (Suc n)) d} ⊆ {d. fst (stage I D E g G n) d} ∪ (λ(ψ, y, xs, Ss). wF I D E (g 0) (G 0) ψ y xs Ss) `
110   (UNIV × UNIV × {xs. set xs ⊆ {d. fst (stage I D E g G n) d}} × {Ss. set Ss ⊆ {S. snd (stage I D E g G n) S}})" (is "⊆ ?R")
111 proof (rule subsetI, simp only: mem_Collect_eq)
112   fix d assume "fst (stage I D E g G (Suc n)) d"
113   then consider "fst (stage I D E g G n) d"
114   | "ψ y xs Ss where "∀x∈set xs. fst (stage I D E g G n) x" "∀S∈set Ss. snd (stage I D E g G n) S"
115     "d = wF I D E (g 0) (G 0) ψ y xs Ss" by auto
116   thus "d ∈ ?R" by cases (auto intro!: rev_image_eqI[where x="(ψ, y, xs, Ss)"])
117 qed
118 lemma snd_stage_Suc_subset:
119   "{S. snd (stage I D E g G (Suc n)) S} ⊆ {S. snd (stage I D E g G n) S} ∪ (λ(ψ, Y, xs, Ss). wS I D E (g 0) (G 0) ψ Y xs Ss) `
120   (UNIV × UNIV × {xs. set xs ⊆ {d. fst (stage I D E g G n) d}} × {Ss. set Ss ⊆ {S. snd (stage I D E g G n) S}})" (is "⊆ ?R")
121 proof (rule subsetI, simp only: mem_Collect_eq)
122   fix S assume "snd (stage I D E g G (Suc n)) S"
123   then consider "snd (stage I D E g G n) S"
124   | "ψ Y xs Ss where "∀x∈set xs. fst (stage I D E g G n) x" "∀S'∈set Ss. snd (stage I D E g G n) S'"
125     "S = wS I D E (g 0) (G 0) ψ Y xs Ss" by auto
126   thus "S ∈ ?R" by cases (auto intro!: rev_image_eqI[where x="(ψ, Y, xs, Ss)"])
127 qed
128 --<Monotonicity of the 'stage' chain: a single-step extension lifts to arbitrary suffixes.>
129 lemma fst_stage_mono: "m ≤ n ⟹ fst (stage I D E g G m) d ⟹ fst (stage I D E g G n) d"
130 by (induct n) (auto simp del: stage.simps simp: stage.simps(2) le_Suc_eq)
131 lemma snd_stage_mono: "m ≤ n ⟹ snd (stage I D E g G m) S ⟹ snd (stage I D E g G n) S"
132 by (induct n) (auto simp del: stage.simps simp: stage.simps(2) le_Suc_eq)
133 --<Finitely many parameters in the omega-union all share a common stage.>
134 lemma common_stage_lifted:
135   assumes hx: "∀x∈set xs. ∃n. fst (stage I D E g G n) x" and hS: "∀S∈set Ss. ∃n. snd (stage I D E g G n) S"
136   shows "∃N. (∀x∈set xs. fst (stage I D E g G N) x) ∧ (∀S∈set Ss. snd (stage I D E g G N) S)"
137 proof -
138   obtain fX where fX: "∀x∈set xs. fst (stage I D E g G (fX x)) x" using hx by metis
139   obtain fS where fS: "∀S∈set Ss. snd (stage I D E g G (fS S)) S" using hS by metis
140   let ?N = "Max (insert 0 (fX ` set xs ∪ fS ` set Ss))"
141   have bX: "fX x ≤ ?N" if "x ∈ set xs" for x using that by (simp add: Max_ge)
142   have bS: "fS S ≤ ?N" if "S ∈ set Ss" for S using that by (simp add: Max_ge)
143   from bX fX fst_stage_mono have "∀x∈set xs. fst (stage I D E g G ?N) x" by blast
144   moreover from bS fS snd_stage_mono have "∀S∈set Ss. snd (stage I D E g G ?N) S" by blast
145   ultimately show ?thesis by blast
146 qed

```

Fig. 17. Theory MS0inHOL.lowenheim_skolem_lemmas.thy (Part 2 of 4): the witness-realisation lemmas wF_realizes/wS_realizes, the per-step Skolem closure skolem_step_FO/skolem_step_SO, the subset book-keeping fst_stage_Suc_subset/snd_stage_Suc_subset, the stage monotonicity fst_stage_mono/snd_stage_mono, and the common-stage lemma common_stage_lifted.

```

147 --<A wF / wS witness built from hull-valued parameters is itself in the hull. Combines common_stage_lifted
148 with the single_Suc_step intros.>
149 lemma wF_in_omega_union:
150   assumes "∀x∈set xs. ∃n. fst (stage I D E g G n) x" and "∀S∈set Ss. ∃n. snd (stage I D E g G n) S"
151   shows "∃n. fst (stage I D E g G n) (wF I D E (g 0) (G 0) ψ y xs Ss)"
152   using common_stage_lifted[OF assms] by (meson fst_stage_Suc_intro)
153 lemma wS_in_omega_union:
154   assumes "∀x∈set xs. ∃n. fst (stage I D E g G n) x" and "∀S∈set Ss. ∃n. snd (stage I D E g G n) S"
155   shows "∃n. snd (stage I D E g G n) (wS I D E (g 0) (G 0) ψ y xs Ss)"
156   using common_stage_lifted[OF assms] by (meson snd_stage_Suc_intro)
157 --<Tarski-Vaught closure of the omega-union: combines 'skolem_step_FO/SO' with 'wF/wS_in_omega_union'.>
158 lemma stage_TV_FO:
159   assumes "a into (λd. ∃n. fst (stage I D E g G n) d)" and "b into (λS. ∃n. snd (stage I D E g G n) S)"
160   and "∃d. D d ∧ (I,D,E).a[y←d].b =d ψ"
161   shows "∃d. (∃n. fst (stage I D E g G n) d) ∧ (I,D,E).a[y←d].b =d ψ"
162   using skolem_step_FO[OF assms(3), of "g 0" "G 0"]
163   wF_in_omega_union[of "map a [0..d ψ"
167   shows "∃S. (∃n. snd (stage I D E g G n) S) ∧ (I,D,E).a,b(Y←S) =d ψ"
168   using skolem_step_SO[OF assms(3), of "g 0" "G 0"]
169   wS_in_omega_union[of "map a [0..

```

Fig. 18. Theory `MSOinHOL_lowenheim_skolem_lemmas.thy` (Part 3 of 4): the ω -union membership lemmas `wF_in_omega_union`/`wS_in_omega_union`, the first- and second-order Tarski–Vaught closures `stage_TV_FO`/`stage_TV_SO`, the inductive countability step `cntDE_step`, the carriers-stay-inside lemmas `fst_stage_subD`/`snd_stage_subE`, stage-wise and ω -countability `stage_countable`/`stage_omega_countable`, and the surjective reindexing `reindex_one`.

```

221 then obtain k where "from_nat_into ?D k = d" by auto
222 moreover have "¬ free (fr + k)" using bound by force
223 ultimately have "g' (fr + k) = d" by (simp add: g'_def)
224 thus "Range g' d" by auto
225 qed
226 moreover have "∧x. free x ⇒ g' x = g x" by (simp add: g'_def)
227 ultimately show ?thesis using that by blast
228 qed
229 —<reindex (both sorts): reindex the (countably many) variables to surject onto the countable @ {text "D0, E0"},
230 while agreeing with the originals on the finitely many free variables of  $\varphi$ . Two applications of 'reindex_one'.>
231 lemma reindex:
232   assumes "countable {d. D0 d}" and "D0 d0" and "countable {S. E0 S}" and "E0 S0"
233   and "Range g ⊆ D0" and "Range G ⊆ E0"
234   obtains g' G' where "Range g' = D0" and "Range G' = E0"
235   and "∧x. x free_in  $\varphi$  ⇒ g' x = g x" and "∧X. X free2_in  $\varphi$  ⇒ G' X = G X"
236 proof -
237   obtain g' where g': "Range g' = D0" "∧x. x free_in  $\varphi$  ⇒ g' x = g x"
238   using reindex_one[OF assms(1,2,5), of "λx. x free_in  $\varphi$ " "fresh  $\varphi$ "] L6 by blast
239   obtain G' where G': "Range G' = E0" "∧X. X free2_in  $\varphi$  ⇒ G' X = G X"
240   using reindex_one[OF assms(3,4,6), of "λX. X free2_in  $\varphi$ " "fresh2  $\varphi$ "] N6 by blast
241   show ?thesis using that g' G' by blast
242 qed
243 —<reindex + coincidence in one step: obtain @ {text "g'/G'"} surjecting onto the countable @ {text "D0/E0"},
244 whose induced truth on  $\varphi$  coincides with the original.>
245 lemma reindex_coincide:
246   assumes "countable {d. D0 d}" and "countable {S. E0 S}" and "Range g ⊆ D0" and "Range G ⊆ E0"
247   obtains g' G' where "Range g' = D0" and "Range G' = E0" and "(⟨(I, D0, E0), g, G⟩ ⊨d  $\varphi$ ) = (⟨(I, D0, E0), g', G'⟩ ⊨d  $\varphi$ )"
248 proof -
249   have nD: "D0 (g 0)" and nE: "E0 (G 0)" using assms(3,4) by auto
250   obtain g' G' where rg: "Range g' = D0" and rG: "Range G' = E0"
251   and ag: "∧x. x free_in  $\varphi$  ⇒ g' x = g x" and aG: "∧X. X free2_in  $\varphi$  ⇒ G' X = G X"
252   using reindex[OF assms(1) nD assms(2) nE assms(3,4)] by blast
253   have "(⟨(I, D0, E0), g, G⟩ ⊨d  $\varphi$ ) = (⟨(I, D0, E0), g', G'⟩ ⊨d  $\varphi$ )" by (rule coincidence) (simp_all add: ag aG)
254   thus thesis using that rg rG by blast
255 qed
256 end

```

Fig. 19. Theory MS0inHOL_lowenheim_skolem_lemmas.thy (Part 4 of 4): the conclusion of reindex_one, the surjective reindexing reindex, and the combined reindex_coincide lemma.

```

1 theory MSOinHOL_lowenheim_skolem imports MSOinHOL_lowenheim_skolem_lemmas
2 begin
3   <Two-sorted elementary substructure (Tarski-Vaught form).>
4   definition ElementarySubstructure ("⊆E") where "(I,D',E') ⊆E (I,D,E) ≡
5     D' ⊆ D ∧ E' ⊆ E ∧ (∀ g G φ. G into E' → g into D' → ((I,D,E),g,G ⊨d φ) = ((I,D',E'),g,G ⊨d φ)))"
6   <Minimal interpretation whose range model is elementary in a larger model.>
7   locale MinS_ES = MinS + fixes DD::D and EE::E
8   assumes ES: "(II,Range gg, Range GG) ⊆E (II,DD,EE)"
9   begin
10    lemma N_valid_ES: "(II,DD,EE),gg,GG ⊨d φ = (II,Range gg, Range GG),gg,GG ⊨d φ"
11    using ES ElementarySubstructure_def by (smt (verit))
12  end
13  <Specialisation to the standard model @ {text "(II,Univ,Univ)"}.>
14  locale MinS_ES_Univ = MinS_ES II gg GG Univ Univ for II gg GG
15  begin
16    lemma N_valid_ES_Univ: "(II,Univ,Univ),gg,GG ⊨d φ = (II,Range gg,Range GG),gg,GG ⊨d φ" using N_valid_ES,
17  end
18  <Range-relative validity: the right-hand side of FaithfulMS_all.>
19  definition RangeValid ("⊨r" 9) where "⊨r φ ≡ ∀! g G. (I, Range g, Range G),g,G ⊨d φ"
20  lemma into_Range: "f into Range f" by auto
21  <Easy direction.>
22  lemma ValD_imp_RangeValid: "⊨d φ ⇒ ⊨r φ" unfolding RangeValid_def ValD_def using into_Range by smt
23  <Truth preservation across a Tarski-Vaught-closed sub-pair @ {text "(D0,E0)"}.>
24  lemma truth_pres:
25    assumes subD: "D0 ⊆ D" and subE: "E0 ⊆ E"
26    and TVfo: "∀ y ψ a b. a into D0 ⇒ b into E0 ⇒ (∃ d. D d ∧ (I,D,E),a[y←d],b ⊨d ψ) ⇒ (∃ d. D0 d ∧ (I,D,E),a[y←d],b ⊨d ψ)"
27    and TVso: "∀ y ψ a b. a into D0 ⇒ b into E0 ⇒ (∃ S. E S ∧ (I,D,E),a,b(Y←S) ⊨d ψ) ⇒ (∃ S. E0 S ∧ (I,D,E),a,b(Y←S) ⊨d ψ)"
28    and aD: "a into D0" and bE: "b into E0"
29    shows "(I,D0,E0),a,b ⊨d ψ = ((I,D,E),a,b ⊨d ψ)" using aD bE
30  proof (induct ψ arbitrary: a b)
31    case (ExD y ψ) thus ?case using ExD.hyps[of "a[y←_]"] TVfo[of a b y ψ] subD by (force simp: EnvMod_def) next
32    case (ExD2 Y ψ) thus ?case using ExD2.hyps[of a "b(Y←_)"] TVso[of a b Y ψ] subE by (force simp: SetMod_def)
33  qed simp_all
34  <Countable seed-form Tarski-Vaught hull: extends any countable nonempty @ {text "(D0,E0) ⊆ (D,E)"}>
35  to a countable TV-closed pair @ {text "(N,M)"} between them.>
36  lemma skolem_hull:
37    fixes I :: "T" and D :: "D" and E :: "E"
38    assumes "D0 ⊆ D" and "E0 ⊆ E" and "∃ x. D0 x" and "∃ S. E0 S" and "countable {d. D0 d}" and "countable {S. E0 S}"
39    obtains N M where "N ⊆ D" and "M ⊆ E" and "D0 ⊆ N" and "E0 ⊆ M"
40    and "countable {d. N d}" and "countable {S. M S}"
41    and "∀ y ψ a b. a into N ⇒ b into M ⇒ (∃ d. D d ∧ (I,D,E),a[y←d],b ⊨d ψ) ⇒ (∃ d. N d ∧ (I,D,E),a[y←d],b ⊨d ψ)"
42    and "∀ y ψ a b. a into N ⇒ b into M ⇒ (∃ S. E S ∧ (I,D,E),a,b(Y←S) ⊨d ψ) ⇒ (∃ S. M S ∧ (I,D,E),a,b(Y←S) ⊨d ψ)"
43  proof -
44    obtain g :: "E" where g: "Range g = D0"
45    by (metis (mono_tags, lifting) Collect_inj assms(3,5) empty_Collect_eq full_SetCompr_eq range_from_nat_into)
46    obtain G :: "G" where G: "Range G = E0"
47    by (metis (mono_tags, lifting) Collect_inj assms(4,6) empty_Collect_eq full_SetCompr_eq range_from_nat_into)
48    define Dh where Dh: "Dh = (λd. ∃ n. fst (stage I D E g G n) d)"
49    define Eh where Eh: "Eh = (λS. ∃ n. snd (stage I D E g G n) S)"
50    have base: "Range g d ⇒ fst (stage I D E g G 0) d" "Range G S ⇒ snd (stage I D E g G 0) S" for d S by simp_all
51    have RgDh: "Range g ⊆ Dh" and RGEh: "Range G ⊆ Eh" using base unfolding Dh Eh by blast+
52    have DhsubD: "Dh ⊆ D" using fst_stage_subD assms(1) unfolding Dh by (metis (full_types) g)
53    have EhsubE: "Eh ⊆ E" using snd_stage_subE assms(2) unfolding Eh by (metis (mono_tags, lifting) G)
54    have cDh: "countable {d. Dh d}" and cEh: "countable {S. Eh S}" unfolding Dh Eh by (rule stage_omega_countable)+
55    have TVfo: "a into Dh ⇒ b into Eh ⇒ (∃ d. D d ∧ (I,D,E),a[y←d],b ⊨d ψ) ⇒ (∃ d. Dh d ∧ (I,D,E),a[y←d],b ⊨d ψ)"
56    for y ψ a b unfolding Dh Eh by (rule stage_TV_FO)
57    have TVso: "a into Dh ⇒ b into Eh ⇒ (∃ S. E S ∧ (I,D,E),a,b(Y←S) ⊨d ψ) ⇒ (∃ S. Eh S ∧ (I,D,E),a,b(Y←S) ⊨d ψ)"
58    for Y ψ a b unfolding Dh Eh by (rule stage_TV_SO)
59    show ?thesis using DhsubD EhsubE G RGEh RgDh TVfo TVso cDh cEh g that by force
60  qed
61  <Downward Löwenheim-Skolem: TV-closure is upgraded to genuine elementarity via @ {text truth_pres}.>
62  theorem DownwardLowenheimSkolem:
63    assumes "D0 ⊆ D" and "E0 ⊆ E" and "∃ x. D0 x" and "∃ S. E0 S" and "countable {d. D0 d}" and "countable {S. E0 S}"
64    shows "∃ N M. D0 ⊆ N ∧ countable {d. N d} ∧ E0 ⊆ M ∧ countable {S. M S} ∧ (I,N,M) ⊆E (I,D,E)"
65  proof -
66    obtain N M where "N ⊆ D" "M ⊆ E" "D0 ⊆ N" "E0 ⊆ M" "countable {d. N d}" "countable {S. M S}"
67    and "∀ y ψ a b. a into N ⇒ b into M ⇒ ∃ d. (I,D,E),a[y←d],b ⊨d ψ ⇒ ∃ d. N d ∧ (I,D,E),a[y←d],b ⊨d ψ"
68    and "∀ y ψ a b. a into N ⇒ b into M ⇒ ∃ S. (I,D,E),a,b(Y←S) ⊨d ψ ⇒ ∃ S. M S ∧ (I,D,E),a,b(Y←S) ⊨d ψ"
69    using skolem_hull[of D0 D E0 E, OF assms] by blast
70    hence "D0 ⊆ N ∧ countable {d. N d} ∧ E0 ⊆ M ∧ countable {S. M S} ∧ (I,N,M) ⊆E (I,D,E)"
71    using ElementarySubstructure_def truth_pres skolem_hull[of D0 D E0 E, OF assms] by auto
72    thus ?thesis by blast
73  qed

```

Fig. 20. Theory MSOinHOL_lowenheim_skolem.thy (Part 1 of 2): the elementary-substructure layer — the relation ElementarySubstructure ($\subseteq_E$) and the locales MinS_ES/MinS_ES_Univ — range-relative validity RangeValid, the easy direction ValD_imp_RangeValid, truth preservation truth_pres for a Tarski–Vaught-closed sub-pair, the countable seed hull skolem_hull, and the first-class downward Löwenheim–Skolem theorem DownwardLowenheimSkolem.

```

74 --<Strongest faithfulness: standard validity = minimal validity over elementary interpretations.>
75 theorem Deep'_to_MinS: "(|=^d φ) = (∀ll gg GG. MinS_ES_Univ ll gg GG → MinS.ValM (MinS.DpToShM ll gg GG φ))"
76 proof
77   assume "(|=^d φ)" thus "∀ll gg. MinS_ES_Univ ll gg ⊆ (λGG. MinS.ValM (MinS.DpToShM ll gg GG φ))"
78   using MinS.FaithfulMD MinS_ES_ℳ_valid_ES MinS_ES_Univ_def ValD'_def by blast
79 next
80   assume A: "∀ll gg GG. MinS_ES_Univ ll gg GG → MinS.ValM (MinS.DpToShM ll gg GG φ)"
81   show "(|=^d φ)" unfolding ValD'_def
82   proof (safe)
83     fix l :: ℒ and g :: ℰ and G :: ℒ
84     have "Range g ⊆ Univ" "Range G ⊆ Univ" "∃x. Range g x" "∃x. Range G x"
85     "countable {d. MSOinHOL_preliminaries.Range g d}" "countable {S. MSOinHOL_preliminaries.Range G S}"
86     by (auto simp add: full_SetCompr_eq)
87     then obtain N M where 7: "Range g ⊆ N" "countable {d. N d}" "Range G ⊆ M" "countable {S. M S}"
88     and ES: "(l, N, M) ⊆ ℰ (l, Univ, Univ)"
89     using DownwardLowenheimSkolem[where D₀="Range g" and E₀="Range G" and D=Univ and E=Univ] by metis
90     then obtain g' where g': "Range g' = N" and G': "Range G' = M" and
91     g'G': "(l, Range g', Range G'), g', G' |=^d φ) = ((l, N, M), g, G |=^d φ)" using reindex by (smt (verit, best) reindex_coincide)
92     hence "(l, Range g', Range G') ⊆ ℰ (l, Univ, Univ)" using ES by presburger
93     then interpret MinS_ES_Univ l g' G' by (simp add: MinS_ES.intro MinS_ES_Univ_def)
94     have "MinS.ValM (MinS.DpToShM l g' G' φ)" using A MinS_ES_Univ_axioms by blast
95     hence "(l, N, M), g, G |=^d φ" using FaithfulMDlem ValM_def g' G' g'G' by blast
96     thus "(l, Univ, Univ), g, G |=^d φ" using ES unfolding ElementarySubstructure_def by (smt (verit, best) "7"(1,3) G' g')
97   qed
98 qed
99 --<Faithfulness to the standard reading (companion of @{text Faithful_to_Henkin}).>
100 corollary Faithful_to_Standard:
101   "(|=^d φ) = (∀ll gg GG. MinS_ES_Univ ll gg GG → MinS.ValM (MinS.DpToShM ll gg GG φ))" using Deep'_to_MinS .
102 --<Earlier Range-seeded route to the general-reading converse: the hull packaged with its
103 truth-preservation payoff, derived in one step from @{text DownwardLowenheimSkolem}.>
104 lemma ls_hull:
105   assumes gD: "g into D" and GE: "G into E"
106   obtains D₀ E₀ where "D₀ ⊆ D" and "E₀ ⊆ E" and "Range g ⊆ D₀" and "Range G ⊆ E₀"
107   and "countable {d. D₀ d}" and "countable {S. E₀ S}" and "(l, D₀, E₀), g, G |=^d φ) = ((l, D, E), g, G |=^d φ)"
108 proof -
109   have *: "Range g ⊆ D" "Range G ⊆ E" "∃x. Range g x" "∃x. Range G x"
110   "countable {d. Range g d}" "countable {S. Range G S}"
111   using assms by (auto simp: full_SetCompr_eq)
112   obtain N M where N: "Range g ⊆ N" "countable {d. N d}" "Range G ⊆ M" "countable {S. M S}"
113   and ES: "(l, N, M) ⊆ ℰ (l, D, E)"
114   using DownwardLowenheimSkolem[where D₀="Range g" and E₀="Range G", of *(1,2,3,4,5,6), of l] by metis
115   have sub: "N ⊆ D" "M ⊆ E" and ES': "λg G φ. G into M ⇒ g into N ⇒ ((l, D, E), g, G |=^d φ) = ((l, N, M), g, G |=^d φ)"
116   using ES unfolding ElementarySubstructure_def by auto
117   have gn: "g into N" "G into M" using into_Range N(1,3) by (metis (mono_tags, lifting))
118   show thesis using that[OF sub N(1,3,2,4) ES'[OF gn(2) gn(1), symmetric]] .
119 qed
120 --<Range reduction: countermodels reduce to range-only countermodels.>
121 lemma Range_reduction:
122   assumes "g into D" and "G into E" obtains g' G' where "(l, Range g', Range G'), g', G' |=^d φ) = ((l, D, E), g, G |=^d φ)"
123 proof -
124   obtain D₀ E₀ where *: "Range g ⊆ D₀" "Range G ⊆ E₀" "countable {d. D₀ d}" "countable {S. E₀ S}"
125   and eq: "(l, D₀, E₀), g, G |=^d φ) = ((l, D, E), g, G |=^d φ)" using assms by (rule ls_hull)
126   obtain g' G' where rg: "Range g' = D₀" and rG: "Range G' = E₀"
127   and coin: "(l, D₀, E₀), g, G |=^d φ) = ((l, D₀, E₀), g', G' |=^d φ)" using reindex_coincide[OF *(3,4,1,2)] by blast
128   show thesis using that[OF g' G' rg rG eq coin] by simp
129 qed
130 --<Range-relative = general (Henkin) deep validity.>
131 theorem RangeValid_imp_ValD: "(|=^r φ) ⇒ (|=^d φ)" unfolding RangeValid_def ValD_def using Range_reduction by metis
132 corollary RangeValid_iff_ValD: "(|=^r φ) = (|=^d φ)" using RangeValid_imp_ValD ValD_imp_RangeValid by blast
133 --<Faithfulness to the general (Henkin) reading: with @{text Faithful_to_Standard} the two
134 faithfulness payoffs, separated by the limitative @{text Standard_strictly_stronger} below.>
135 corollary Faithful_to_Henkin: "(|=^r φ) = (|=^d φ)" using RangeValid_iff_ValD .
136 --<The standard reading is strictly stronger; comprehension witnesses the gap.>
137 lemma Standard_strictly_stronger: "(∀(φ::ℱ). (|=^d φ) → (|=^r φ)) ∧ (∃(φ::ℱ). (|=^d φ) ∧ ¬ (|=^r φ))"
138 proof
139   show "∀(φ::ℱ). (|=^d φ) → (|=^r φ)" using Val by blast next
140   let ?φ = "∃d:(0::V2). ∀d:(0::V). (((0::V2) d)((0::V))) ⇐⇒ d ((0::R) d)((0::V), (0::V)))"
141   have "(|=^d ?φ)" by (rule comprehension_atom)
142   moreover have "¬ (|=^r ?φ)"
143     unfolding DefD apply (rule notI, drule spec[where x="λr (a::D) b. a = b"], drule spec[where x="Univ::D⇒bool"],
144     drule spec[where x="λS::D⇒bool. ∃d::D. ¬S d"], drule spec[where x="λx::V. undefined::D"],
145     drule spec[where x="λX::V2. λd::D. False"]) by (auto simp: SetMod_def EnvMod_def)
146   ultimately show "∃(φ::ℱ). (|=^d φ) ∧ ¬ (|=^r φ)" by blast
147 qed
148 end

```

Fig. 21. Theory `MSOinHOL_lowenheim_skolem.thy` (Part 2 of 2): the strongest faithfulness theorem `Deep'_to_MinS` (standard validity $\Leftrightarrow$ minimal validity over elementary interpretations) with its corollary `Faithful_to_Standard`; then the earlier general-reading route, derived in one step from `DownwardLowenheimSkolem` — the assembled elementary-substructure hull `ls_hull`, the range reduction `Range_reduction`, the main theorem `RangeValid_imp_ValD` with corollaries `RangeValid_iff_ValD`/`Faithful_to_Henkin`, and the negative result `Standard_strictly_stronger` with its explicit diagonal countermodel.

```

1 theory MS0inHOL_shallow_minimal_elementary imports MS0inHOL_faithfulness_locale MS0inHOL_lowenheim_skolem
2 begin
3 <Extra simp rules for @{term DpToShS} (derived quantifiers and connectives).>
4 lemma (in MinS) DpToShS_All[simp]: "( $\forall^d x. \varphi$ ) = ( $\forall^m d. ([x \leftarrow d](\varphi))$ )"
5 unfolding DefD DefM by (simp add: ExM_def NegM_def ren_subst_def)
6 lemma (in MinS) DpToShS_All2[simp]: "( $\forall^d_2 x. \varphi$ ) = ( $\forall^m_2 d. ([x \leftarrow r_2 d](\varphi))$ )"
7 unfolding DefD DefM using MinS.FaithfulMDlem by auto
8 lemma (in MinS) DpToShS_Equiv[simp]: "( $\varphi \longleftrightarrow^d \psi$ ) = ( $(\varphi \longleftrightarrow^m \psi)$ )"
9 unfolding DefD DefM by (simp add: AndM_def NegM_def)
10 lemma (in MinS) DpToShS_Imp[simp]: "( $\varphi \supset^d \psi$ ) = ( $(\varphi) \supset^m (\psi)$ )"
11 unfolding DefD DefM by (simp add: AndM_def NegM_def)
12 <Elementary constants-only presentation: @{text II}, @{text gg}, @{text GG} chosen so that the
13 global interpretation is a @{text MinS_ES_Univ} (existence via @{text Deep'_to_MinS}).>
14 consts II::T gg::E GG::G
15 specification (II gg GG) ES_Univ: "MinS_ES_Univ II gg GG" by (meson Deep'_to_MinS RelativeTruthD.simps ValD'_def)
16 global_interpretation MinS_ES_Univ II gg GG using ES_Univ by auto
17 notation AtmM ("_m'(_)'") and PrdM ("_m'(_)'") and NegM ("_m'_" [58]59)
18 and AndM (infix " $\wedge^m$ " 56) and ExM (binder " $\exists^m$ " 53) and ExM2 (binder " $\exists^m_2$ " 53)
19 and OrM (infix " $\vee^m$ " 54) and ImpM (infix " $\supset^m$ " 55) and IffM (infix " $\longleftrightarrow^m$ " 54)
20 and AllM (binder " $\forall^m$ " 53) and AllM2 (binder " $\forall^m_2$ " 53) and ValM ("_m'" 9) and DpToShM ("_")
21 <Faithfulness under universal carriers.>
22 lemma FaithfulMD_ES: "( $\models^m \varphi$ ) = (II,Univ,Univ).gg.GG  $\models^d \varphi$ " using FaithfulMD  $\wedge$ _valid_ES_Univ by blast
23 <Inheritance: standard validity transfers to minimal validity.>
24 lemma ValidM_if_ValidD': "( $\models^d \varphi$ )  $\implies$  ( $\models^m \varphi$ )" by (metis MinS_ES_Univ_axioms Deep'_to_MinS)
25 end
    
```

Fig. 22. Theory `MS0inHOL_shallow_minimal_elementary.thy`: an *elementary* constants-only presentation of the minimal shallow embedding — four `DpToShS` simplification lemmas for the derived quantifiers and connectives (`DpToShS_All`, `DpToShS_All2`, `DpToShS_Equiv`, `DpToShS_Imp`), the three uninterpreted constants `II`, `gg`, `GG`, a *specification* establishing the existence of a global interpretation that forms a `MinS_ES_Univ` elementary substructure (with existence justified by `Deep'_to_MinS`), the corresponding *global_interpretation* re-issuing the locale notation at the constants level, and the inheritance lemmas `FaithfulMD_ES` (faithfulness under universal carriers) and `ValidM_if_ValidD'` (standard validity lifts to minimal validity).

```

1 theory MSOinHOL_experiments_classic_elementary
2 imports MSOinHOL_deep MSOinHOL_shallow MSOinHOL_shallow_minimal_elementary
3 begin
4 <Extra simp rules for derived quantifiers / connectives.>
5 lemma ren_for_subst2_simp_All[simp]: "ren_for_subst2 X Z (∀d2Y. φ) = (if Y = Z ∧ X free2_in φ
6   then let f = max (fresh2 φ) (Z+1); φ' = [Y ←2f](φ) in (∀d2f. ren_for_subst2 X Z φ')
7   else ∀d2Y. ren_for_subst2 X Z φ)"
8 unfolding DefD by (auto simp: Let_def)
9 lemma subst2_all[simp]: "[X ←2Z](∀d2Y. φ) = (if X = Y then (∀d2Y. φ) else (∀d2Y. [X ←2Z](φ)))"
10 unfolding DefD by (auto simp: Let_def)
11 lemma free2_in_equiv[simp]: "X free2_in (φ ←d ψ) = (X free2_in φ ∨ X free2_in ψ)" by (auto simp add: DefD)
12 lemma free2_in_all[simp]: "X free2_in (∀d2Y. φ) = (X free2_in φ)" by (simp add: AllD_def)
13 lemma free2_in_all2[simp]: "X free2_in (∀d2Y. φ) = (X free2_in φ ∧ X ≠ Y)" by (metis AllD2_def is_free2_simps)
14 <Some abbreviations for variables.>
15 abbreviation "(x::V) ≡ 1" abbreviation "(y::V) ≡ 2" abbreviation "(z::V) ≡ 3"
16 abbreviation "(X::V2) ≡ 1" abbreviation "(Y::V2) ≡ 2" abbreviation "(Z::V2) ≡ 3"
17 abbreviation "(u::V) ≡ 4" abbreviation "(v::V) ≡ 5" consts P::R
18 <Boolean closure (Büchi 1960; Thomas 1997) under @{text "≡d"}: witnesses supplied via @{text ex1}.>
19 lemma complement_d: "≡d (∀d2X. ∃d2Z. ∀dx. (Zd(x) ←d ¬ Xd(x)))"
20 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
21 subgoal for S by (rule exI[of _ "λd. ¬ S d"]) (auto simp: SetMod_def EnvMod_def) done
22 lemma intersection_d: "≡d (∀d2X. ∀d2Y. ∃d2Z. ∀dx. (Zd(x) ←d (Xd(x) ∧d Yd(x))))"
23 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
24 subgoal for S Sa by (rule exI[of _ "λd. S d ∧ Sa d"]) (auto simp: SetMod_def EnvMod_def) done
25 lemma intersection_d': "≡d (∃d2Z. ∀dx. (Zd(x) ←d (Xd(x) ∧d Yd(x))))"
26 by (simp add: comprehension_schema)
27 lemma union_d: "≡d (∀d2X. ∀d2Y. ∃d2Z. ∀dx. (Zd(x) ←d (Xd(x) ∨d Yd(x))))"
28 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
29 subgoal for S Sa by (rule exI[of _ "λd. S d ∨ Sa d"]) (auto simp: SetMod_def EnvMod_def) done
30 <Graph operations (Courcelle 2012).>
31 lemma separation_d: "≡d (∀d2X. ∃d2Z. ∀dx. (Zd(x) ←d (Xd(x) ∧d Pd(x,x))))"
32 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
33 subgoal for I S by (rule exI[of _ "λd. S d ∧ I P d d"]) (auto simp: SetMod_def EnvMod_def) done
34 lemma image_d: "≡d (∀d2X. ∃d2Y. ∀dx. (Yd(x) ←d ∃dy. (Xd(y) ∧d Pd(y,x))))"
35 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
36 subgoal for I S by (rule exI[of _ "λd. ∃d'. S d' ∧ I P d' d"]) (auto simp: SetMod_def EnvMod_def) done
37 lemma preimage_d: "≡d (∀d2X. ∃d2Y. ∀dx. (Yd(x) ←d ∃dy. (Pd(x,y) ∧d Xd(y))))"
38 unfolding DefD apply (intro allI, simp add: SetMod_def EnvMod_def, intro allI)
39 subgoal for I S by (rule exI[of _ "λd. ∃d'. I P d d' ∧ S d'"]) (auto simp: SetMod_def EnvMod_def) done
40 <Reachability (Basin & Klarlund 1995): not universally valid; reflexive variant is.>
41 lemma reachability_not_valid_d: "≡d (∀d2Z. ((Zd(x) ∧d (∀du. (Zd(u) ⊃d Pd(u,v) ⊃d Zd(v)))) ⊃d Zd(y)))"
42 unfolding DefD apply simp nitpick oops
43 lemma reachability_reflexive_d: "≡d (∀d2Z. ((Zd(x) ∧d (∀du. (Zd(u) ⊃d Pd(u,v) ⊃d Zd(v)))) ⊃d Zd(x)))"
44 unfolding DefD by simp
45 <2-colorability (Thomas 1997): refuted on $K_{3,5}$.>
46 lemma two_colorability_not_valid_d: "≡d (∃d2Z. ∀dx. ∀dy. (Pd(x,y) ⊃d (Zd(x) ←d ¬ Zd(y))))"
47 unfolding DefD apply simp nitpick oops

```

Fig. 23. Theory `MSOinHOL_experiments_classic_elementary.thy` (Part 1 of 2): additional simplification rules for the derived second-order quantifier and equivalence (`ren_for_subst2_simp_All`, `subst2_all`, `free2_in_equiv`, `free2_in_all`, `free2_in_all2`), the convenient variable abbreviations, and the full set of deep-embedding classical MSO landmarks — Boolean closure (`complement_d`, `intersection_d`/`intersection_d'`, `union_d`), monadic graph operations (`separation_d`, `image_d`, `preimage_d`), reachability (`reachability_not_valid_d`, `reachability_reflexive_d`), and 2-colorability (`two_colorability_not_valid_d`).

```

47 unfolding DefD apply simp nitpick oops
48 <Same landmarks in the maximal-shallow embedding: structurally identical proofs.>
49 lemma complement_s: "⊨s (∀s2X. ∃s2Z. ∀sx. (Zs(x) ↔s ¬s Xs(x)))"
50 unfolding DefS apply (intro allI, simp, intro allI) subgoal for S by (rule exI[of _ "λd. ¬ S d"]) auto done
51 lemma intersection_s: "⊨s (∀s2X. ∀s2Y. ∃s2Z. ∀sx. (Zs(x) ↔s (Xs(x) ∧s Ys(x))))"
52 unfolding DefS apply (intro allI, simp, intro allI) subgoal for S Sa by (rule exI[of _ "λd. S d ∧ Sa d"]) auto done
53 lemma union_s: "⊨s (∀s2X. ∀s2Y. ∃s2Z. ∀sx. (Zs(x) ↔s (Xs(x) ∨s Ys(x))))"
54 unfolding DefS apply (intro allI, simp, intro allI) subgoal for S Sa by (rule exI[of _ "λd. S d ∨ Sa d"]) auto done
55 lemma separation_s: "⊨s (∀s2X. ∃s2Z. ∀sx. (Zs(x) ↔s (Xs(x) ∧s Ps(x,x))))"
56 unfolding DefS apply (intro allI, simp, intro allI) subgoal for I S by (rule exI[of _ "λd. S d ∧ I P d d"]) auto done
57 lemma image_s: "⊨s (∀s2X. ∃s2Y. ∀sx. (Ys(x) ↔s (∃sy. (Xs(y) ∧s Ps(y,x))))"
58 unfolding DefS apply (intro allI, simp, intro allI) subgoal for I S by (rule exI[of _ "λd. ∃d'. S d' ∧ I P d' d"]) auto done
59 lemma preimage_s: "⊨s (∀s2X. ∃s2Y. ∀sx. (Ys(x) ↔s (∃sy. (Ps(x,y) ∧s Xs(y))))"
60 unfolding DefS apply (intro allI, simp, intro allI) subgoal for I S by (rule exI[of _ "λd. ∃d'. I P d d' ∧ S d"]) auto done
61 lemma reachability_not_valid_s: "⊨s (∀s2Z. ((Zs(x) ∧s (∀su. (Zs(u) ⊃s (∀sv. (Ps(u,v) ⊃s Zs(v)))))) ⊃s Zs(x)))"
62 unfolding DefS apply (intro allI) apply simp nitpick oops
63 lemma reachability_reflexive_s: "⊨s (∀s2Z. ((Zs(x) ∧s (∀su. (Zs(u) ⊃s (∀sv. (Ps(u,v) ⊃s Zs(v)))))) ⊃s Zs(x)))"
64 unfolding DefS by simp
65 lemma two_colorability_not_valid_s: "⊨s (∃s2Z. ∀sx. ∀sy. (Ps(x,y) ⊃s (Zs(x) ↔s ¬s Zs(y))))"
66 unfolding DefS apply (intro allI) apply simp nitpick oops
67 <Transfer to the elementary minimal embedding via @{text DpToShS} normal-form and @{text ValidM_if_ValidD'}.>
68 lemma compl_m_DpToSh: "(⊨d2X. ∃d2Z. ∀dx. (Zd(x) ↔d ¬d Xd(x))) = (⊨m2X. ∃m2Z. ∀mx. (Zm(x) ↔m ¬m Xm(x)))"
69 by (auto simp add: ren_subst2_def ren_subst_def Let_def) (auto simp: DefD DefM ren_subst_def)
70 lemma compl_m_valid: "⊨m (⊨m2X. ∃m2Z. ∀mx. (Zm(x) ↔m ¬m Xm(x)))"
71 using compl_m_DpToSh ValidM_if_ValidD'[OF complement_d] by simp
72 lemma inters_m_DpToSh: "(⊨d2Z. ∀dx. (Zd(x) ↔d (Xd(x) ∧d Yd(x)))) = (⊨m2Z. ∀mx. (Zm(x) ↔m (Xm(x) ∧m Ym(x))))"
73 by (auto simp add: ren_subst2_def ren_subst_def Let_def) (auto simp: DefD DefM ren_subst_def)
74 lemma DpToSh_allegI: "(⊨d d . ((X ↔d d) (φ))) = ψ d ⇒ (⊨d2X. φ) = (⊨m2X. ψ X)" by simp
75 lemma all_m_eqI: "(⊨d X . φ X = ψ X) ⇒ (⊨m2X . φ X) = (⊨m2X . ψ X)" by presburger
76 lemmas DpToSh_simps = DefM Let_def ren_subst_def DefD ren_subst2_def
77 lemma intersect_m_DpToSh: "(⊨d2X. ∀d2Y. ∃d2Z. ∀dx. (Zd(x) ↔d (Xd(x) ∧d Yd(x)))) = (⊨m2X. ∀m2Y. ∃m2Z. ∀mx. (Zm(x) ↔m (Xm(x) ∧m Ym(x))))"
78 by (auto simp add: DpToSh_simps)
79 apply (rule DpToSh_allegI, simp only: ren_subst2_def subst2_all ren_for_subst2_simp_All, simp)
80 apply (case_tac "d = Z", simp, rule all_m_eqI)
81 subgoal by (auto simp: DpToSh_simps)
82 apply (simp, case_tac "d = Y", simp)
83 subgoal by (auto simp: DpToSh_simps) metis+
84 apply (simp, rule all_m_eqI, auto simp: ren_subst2_def ren_subst_def Let_def, auto simp: DpToSh_simps)
85 by (smt (verit, del_insts) Suc_n_not_le_n max.cobounded1 max.left_commute numeral_2_eq_2 numeral_3_eq_3)
86 lemma intersection_m_valid: "⊨m (⊨m2X. ∀m2Y. ∃m2Z. ∀mx. (Zm(x) ↔m (Xm(x) ∧m Ym(x))))"
87 using ValM_def ValidM_if_ValidD' intersection_d intersect_m_DpToSh by blast
88 <Reachability and 2-colorability remain refuted in the elementary minimal embedding.>
89 lemma reachability_not_valid_m: "⊨m (⊨m2Z. ((Zm(x) ∧m (∀mu. (Zm(u) ⊃m (∀mv. (Pm(u,v) ⊃m Zm(v)))))) ⊃m Zm(x)))"
90 unfolding DefM oops
91 lemma reachability_reflexive_m: "⊨m (⊨m2Z. ((Zm(x) ∧m (∀mu. (Zm(u) ⊃m (∀mv. (Pm(u,v) ⊃m Zm(v)))))) ⊃m Zm(x)))"
92 unfolding DefM by simp
93 lemma two_colorability_not_valid_m: "⊨m (⊨m2Z. ∀mx. ∀my. (Pm(x,y) ⊃m (Zm(x) ↔m ¬m Zm(y))))"
94 unfolding DefM oops
95 end

```

Fig. 24. Theory `MS0inHOL_experiments.classic.elementary.thy` (Part 2 of 2): the same landmarks in the maximal-shallow embedding (the `deep↔maximal-shallow` scripts are line-for-line identical because the goals are HOL-equal after unfolding), and their lifts to the elementary global interpretation — the `DpToShS`-normal-form lemmas (`compl_m_DpToSh`, `inters_m_DpToSh'`, `intersect_m_DpToSh`), the helper combinators (`DpToSh_allegI`, `all_m_eqI`, `DpToSh_simps`), and the `ValidM_if_ValidD'`-style validity transfers (`compl_m_valid`, `intersection_m_valid`); reachability and 2-colorability remain refuted in the elementary minimal embedding as expected.