

Monadic Second-Order Logic in HOL: Deep and Shallow Embeddings with Automated Faithfulness (Isabelle/HOL dataset)

Christoph Benz Müller Daniel Kirchner

July 14, 2026

Abstract

We develop, in Isabelle/HOL, three embeddings of monadic second-order logic (MSO) into classical higher-order logic side by side: a *deep* embedding (an inductive datatype with an explicitly defined satisfaction relation), a *maximal-shallow* embedding, and a *minimal-shallow* embedding — the last a locale parametrised by an interpretation and by first- and second-order assignments. The enabling ingredient is a *two-sorted* capture-avoiding substitution apparatus (predication, substitution, alphabetic renaming, and a substitution lemma, developed once for each of the two binder namespaces), in which each binder is transparent for the other. Faithfulness of all three embeddings is mechanised and automated.

The central result is a fully mechanised two-sorted downward Löwenheim–Skolem theorem. Its corollary identifies range-relative validity with the general (Henkin-style) reading of MSO, while comprehension witnesses that the standard reading is strictly stronger; an elementary-substructure refinement recovers the standard reading from the minimal embedding as well, so the two readings become one device under two interpretation classes (all interpretations versus the elementary ones). We further exercise the embeddings on classical MSO landmarks: Boolean closure and graph operations hold under the standard (full second-order) reading and fail under the general one, making the dichotomy concrete, while reachability and 2-colorability are non-theorems, refuted throughout with `nitpick` countermodels.

Contents

1	Overview	3
2	Preliminaries	4

3	The deep embedding	6
4	Two-sorted substitution	8
4.1	First-order machinery (Part A)	8
4.2	Second-order machinery (Part B)	12
4.3	Auxiliary renaming results	15
5	Shallow embeddings	16
5.1	Maximal-shallow embedding	16
5.2	Minimal-shallow embedding (locale)	18
5.3	Minimal-shallow embedding (translation)	19
6	Faithfulness	20
6.1	The faithfulness locale	20
6.2	Faithfulness theorems	21
7	Comprehension	22
8	Downward Löwenheim–Skolem	23
8.1	Tarski–Vaught stage lemmas	23
8.2	The theorem and its corollaries	34
9	Experiments	41
9.1	Basic landmarks	41
9.2	Classical MSO landmarks	43
9.3	Locale-based variants	47
9.4	Elementary minimal embedding (setup)	47
9.5	Elementary minimal embedding (landmarks)	48
9.6	Additional landmarks	54

1 Overview

This entry extends the deep-and-shallow embedding methodology of *Faithful Logic Embeddings in HOL — Deep and Shallow* [6] from propositional modal logic to a logic with genuine quantifier binding. The shallow-embedding approach goes back to the modal-logic embeddings of Benzmüller and Paulson [4, 5] and underlies the LogiKEy methodology [3]. The background on MSO and its model theory follows the standard references [12, 8, 7]; the notions of general (Henkin) and standard second-order semantics follow [9, 11, 14, 10]. Locales [1] parametrise the minimal embedding throughout. Within the AFP, MSO on words has been treated algorithmically by Traytel and Nipkow [13], who verify decision procedures based on derivatives of regular expressions; the present entry is complementary, being concerned not with decision procedures but with faithful embeddings of MSO into HOL and with the resulting model theory (the general versus standard readings and a two-sorted Löwenheim–Skolem theorem).

The development is organised as follows.

Syntax and the deep embedding. `MSOinHOL_preliminaries` fixes the variable namespaces, domains, and assignment-update operations. `MSOinHOL_deep` introduces MSO formulae as an inductive datatype together with the deep satisfaction relation and the global validity notions $\models^d$ (general; all interpretations) and $\models^{d'}$ (standard; full second-order domain).

Shallow embeddings. `MSOinHOL_shallow` gives the maximal-shallow embedding. `MSOinHOL_shallow_minimal_locale` and `MSOinHOL_shallow_minimal` give the minimal-shallow embedding as a locale `MinS` parametrised by an interpretation and two assignments, together with its translation into HOL.

Substitution. `MSOinHOL_deep_subst_lemma` develops the two-sorted capture-avoiding substitution and renaming machinery and the substitution lemmas `SubstitutionLemma` and `SubstitutionLemma2`, one per binder namespace; `MSOinHOL_subst_extras` collects auxiliary renaming-preservation results.

Faithfulness. `MSOinHOL_faithfulness_locale` proves the faithfulness theorems relating the deep, maximal-shallow and minimal-shallow embeddings (`FaithfulSD`, `FaithfulMD`, `FaithfulMS`, and `FaithfulMS_all`, which identifies minimal validity with range-relative deep validity).

Comprehension and Löwenheim–Skolem. `MSOinHOL_comprehension` isolates the comprehension schema that separates the readings. `MSOinHOL_`

`lowenheim_skolem_lemmas` builds the two-sorted Tarski–Vaught stage construction, and `MSOinHOL_lowenheim_skolem` assembles the headline results: the downward Löwenheim–Skolem theorem `DownwardLowenheimSkolem`, the elementary-substructure relation `ElementarySubstructure`, the faithfulness corollaries `Faithful_to_Henkin` and `Faithful_to_Standard`, and the limitative result `Standard_strictly_stronger`.

Experiments. The `experiments` theories exercise classical MSO landmarks across the embeddings — Boolean closure and graph operations hold under the standard reading and fail under the general one, while reachability [2] and 2-colorability [12] are non-theorems, refuted throughout — using `nitpick` for the distinguishing countermodels; the `_elementary` theories carry the same landmarks over the elementary minimal interpretation.

Disclosure on the use of generative AI. The authors used generative-AI assistants (Anthropic’s Claude family of models) as a drafting aid: to help condense prose, to suggest and shorten selected proofs (in particular parts of the two-sorted Löwenheim–Skolem construction), and to help keep the L^AT_EX sources of this entry tidy. All mathematical content, the theory development, and the design decisions are the authors’ own. Every definition, proof, and piece of text in the final entry has been checked by the authors, who take full responsibility for the content.

2 Preliminaries

```
theory MSOinHOL-preliminaries
  imports Main
begin
```

Some global settings.

```
nitpick-params [user-axioms, expect = genuine]
```

Declarations shared between the deep, maximal-shallow, and minimal-shallow embeddings of MSO in HOL.

<code>typedecl</code> D	— Domain of individuals.
<code>type-synonym</code> $R = nat$	— Binary relation symbols.
<code>type-synonym</code> $V = nat$	— First-order variable symbols.
<code>type-synonym</code> $V2 = nat$	— Second-order variable symbols.
<code>type-synonym</code> $\mathcal{R} = D \Rightarrow D \Rightarrow bool$	— Binary relations on individuals.
<code>type-synonym</code> $\mathcal{I} = R \Rightarrow \mathcal{R}$	— Interpretations of relation symbols.
<code>type-synonym</code> $\mathcal{E} = V \Rightarrow D$	— First-order variable assignments.
<code>type-synonym</code> $\mathcal{G} = V2 \Rightarrow (D \Rightarrow bool)$	— Second-order variable assignments.
<code>type-synonym</code> $\mathcal{D} = D \Rightarrow bool$	— First-order domain restrictions.
<code>type-synonym</code> $\mathcal{P} = (D \Rightarrow bool) \Rightarrow bool$	— Second-order domain restrictions.

Pointwise update of first-order variable assignments.

definition $EnvMod :: \mathcal{E} \Rightarrow V \Rightarrow D \Rightarrow \mathcal{E} \quad (-[- \leftarrow -] [110, 0, 0] 110)$
where $g[x \leftarrow d] \equiv \lambda z. \text{ if } z = x \text{ then } d \text{ else } g \ z$

Pointwise update of second-order variable assignments (distinct notation, angle brackets).

definition $SetMod :: \mathcal{G} \Rightarrow V2 \Rightarrow (D \Rightarrow bool) \Rightarrow \mathcal{G}$
 $(-\langle - \leftarrow - \rangle [110, 0, 0] 110)$
where $G\langle X \leftarrow S \rangle \equiv \lambda Z. \text{ if } Z = X \text{ then } S \text{ else } G \ Z$

Standard lemmas about first-order variable-assignment update.

lemma $L1$ $[simp]: x \neq y \implies (g[y \leftarrow d]) \ x = g \ x$
by $(simp \ add: EnvMod-def)$

lemma $L2: a \neq c \implies g[a \leftarrow d_1][c \leftarrow d_2] = g[c \leftarrow d_2][a \leftarrow d_1]$
by $(auto \ simp: EnvMod-def)$

lemma $L3$ $[simp]: (g[a \leftarrow d]) \ a = d$
by $(simp \ add: EnvMod-def)$

lemma $L4$ $[simp]: g[a \leftarrow d_1][a \leftarrow d_2] = g[a \leftarrow d_2]$
by $(auto \ simp: EnvMod-def)$

Standard lemmas about second-order variable-assignment update.

lemma $M1$ $[simp]: X \neq Y \implies (G\langle Y \leftarrow S \rangle) \ X = G \ X$
by $(simp \ add: SetMod-def)$

lemma $M2: A \neq C \implies G\langle A \leftarrow S_1 \rangle \langle C \leftarrow S_2 \rangle = G\langle C \leftarrow S_2 \rangle \langle A \leftarrow S_1 \rangle$
by $(auto \ simp: SetMod-def)$

lemma $M3$ $[simp]: (G\langle A \leftarrow S \rangle) \ A = S$
by $(simp \ add: SetMod-def)$

lemma $M4$ $[simp]: G\langle A \leftarrow S_1 \rangle \langle A \leftarrow S_2 \rangle = G\langle A \leftarrow S_2 \rangle$
by $(auto \ simp: SetMod-def)$

Bounded quantifiers: $\forall x:D. \varphi$ stands for $\forall x. \ D \ x \longrightarrow \varphi \ x$ and $\exists x:D. \varphi$ stands for $\exists x. \ D \ x \wedge \varphi \ x$.

abbreviation $Ball \ D \ \varphi \equiv \forall x. \ D \ x \longrightarrow \varphi \ x$

syntax $Ball :: pttrn \Rightarrow logic \Rightarrow bool \Rightarrow bool$
 $((\exists \forall (-/:-) ./-) [0, 0, 10] 10)$

translations $\forall x:D. \varphi \rightleftharpoons CONST \ Ball \ D \ (\lambda x. \ \varphi)$

abbreviation $BEx \ D \ \varphi \equiv \exists x. \ D \ x \wedge \varphi \ x$

syntax $BEx :: pttrn \Rightarrow logic \Rightarrow bool \Rightarrow bool$
 $((\exists \exists (-/:-) ./-) [0, 0, 10] 10)$

translations $\exists x:D. \varphi \rightleftharpoons CONST \ BEx \ D \ (\lambda x. \ \varphi)$

Set-as-predicate operations, range, and the universal domain (polymorphic; used for both sorts).

abbreviation *Into* (**infix** *into* 100)
where $f \text{ into } D \equiv \forall x. D (f x)$

abbreviation *Range*
where $\text{Range } f \equiv \lambda x. \exists y. x = f y$

abbreviation *Onto* (**infix** *onto* 100)
where $f \text{ onto } D \equiv D = \text{Range } f$

abbreviation *Subset* (**infix** $\subseteq$ 100)
where $A \subseteq B \equiv \forall x. A x \longrightarrow B x$

abbreviation *Union* (**infixl** $\cup$ 110)
where $A \cup B \equiv \lambda x. A x \vee B x$

abbreviation *Univ*
where $\text{Univ} \equiv \lambda x. \text{True}$

Surjectivity onto the universal predicate coincides with HOL surjectivity.

lemma *onto-Univ*: $\text{surj } f = (f \text{ onto } \text{Univ})$
by (*auto simp: surj-def fun-eq-iff*)

Backward implication; useful for stating equivalences in two directions.

abbreviation *Bimp* (**infixr** $\longleftarrow$ 50)
where $\varphi \longleftarrow \psi \equiv \psi \longrightarrow \varphi$

end

3 The deep embedding

theory *MSOinHOL-deep*
imports *MSOinHOL-preliminaries*
begin

Deep embedding of monadic second-order logic (MSO) in HOL.

Two binders: $\exists^d$ ranges over individuals; $\exists^d_2$ ranges over monadic predicates. $X^d(x)$ is the monadic predication atom “predicate X holds of individual x ”.

datatype $\mathcal{F} =$

$\text{AtmD } R \ V \ V$	$(\neg^d(-), \neg^d)$	— relational atom $r(x, y)$
$\text{PrdD } V_2 \ V$	$(\neg^d(-))$	— monadic predication atom $X(x)$
$\text{NegD } \mathcal{F}$	$(\neg^d - [58] \ 59)$	— negation
$\text{AndD } \mathcal{F} \ \mathcal{F}$	(infixr $\wedge^d$ 56)	— conjunction
$\text{ExD } V \ \mathcal{F}$	$(\exists^d -, -)$	— first-order existential
$\text{ExD2 } V_2 \ \mathcal{F}$	$(\exists^d_2 -, -)$	— second-order (monadic) existential

Further connectives as derived definitions.

definition *OrD* (**infixr** $\vee^d$ 54)

where $\varphi \vee^d \psi \equiv \neg^d(\neg^d \varphi \wedge^d \neg^d \psi)$

definition *ImpD* (**infixr** $\supset^d$ 55)

where $\varphi \supset^d \psi \equiv \neg^d \varphi \vee^d \psi$

definition *IffD* (**infixr** $\longleftrightarrow^d$ 54)

where $\varphi \longleftrightarrow^d \psi \equiv (\varphi \supset^d \psi) \wedge^d (\psi \supset^d \varphi)$

definition *AllD* ($\forall^d$ - .)

where $\forall^d x. \varphi \equiv \neg^d(\exists^d x. \neg^d \varphi)$

definition *AllD2* ($\forall^d_2$ - .)

where $\forall^d_2 X. \varphi \equiv \neg^d(\exists^d_2 X. \neg^d \varphi)$

Relative truth in a model. The first-order existential ranges over the individual domain D ; the second-order existential ranges over the collection E of admissible monadic sets.

primrec *RelativeTruthD* :: $\mathcal{I} \Rightarrow \mathcal{D} \Rightarrow \mathcal{P} \Rightarrow \mathcal{E} \Rightarrow \mathcal{G} \Rightarrow \mathcal{F} \Rightarrow \text{bool}$

$(\langle -, -, \rangle, -, - \models^d - [10, 10, 10, 10, 10] \ 100)$

where

$\langle I, D, E \rangle, g, G \models^d (r^d(x, y)) = I \ r \ (g \ x) \ (g \ y)$
 $\mid \langle I, D, E \rangle, g, G \models^d (X^d(x)) = (G \ X) \ (g \ x)$
 $\mid \langle I, D, E \rangle, g, G \models^d (\neg^d \varphi) = (\neg \langle I, D, E \rangle, g, G \models^d \varphi)$
 $\mid \langle I, D, E \rangle, g, G \models^d (\varphi \wedge^d \psi) = (\langle I, D, E \rangle, g, G \models^d \varphi \wedge \langle I, D, E \rangle, g, G \models^d \psi)$
 $\mid \langle I, D, E \rangle, g, G \models^d (\exists^d x. \varphi) = (\exists d:D. \langle I, D, E \rangle, g[x \leftarrow d], G \models^d \varphi)$
 $\mid \langle I, D, E \rangle, g, G \models^d (\exists^d_2 X. \varphi) = (\exists S:E. \langle I, D, E \rangle, g, G \langle X \leftarrow S \rangle \models^d \varphi)$

Validity: true in every model, every domain-respecting first- and second-order assignment.

definition *ValD* ($\models^d$ - 9)

where $\models^d \varphi \equiv \forall I \ D \ E \ g \ G. \ g \ \text{into} \ D \longrightarrow G \ \text{into} \ E \longrightarrow \langle I, D, E \rangle, g, G \models^d \varphi$

Auxiliary “full-domain” notion of validity: assignments range over the full types (*Univ* on D and on $D \Rightarrow \text{bool}$).

definition *ValD'* ($\models^{d''}$ - 9)

where $\models^{d'} \varphi \equiv \forall I \ g \ G. \ \langle I, \text{Univ}, \text{Univ} \rangle, g, G \models^d \varphi$

General validity implies full-domain validity.

lemma *Val*: $\models^d \varphi \implies \models^{d'} \varphi$

using *ValD'-def* *ValD-def* **by** *simp*

Bag of definitions for collective unfolding.

named-theorems *DefD*

lemmas [*DefD*] = *OrD-def* *ImpD-def* *IffD-def* *AllD-def* *AllD2-def* *ValD-def* *ValD'-def*

end

4 Two-sorted substitution

theory *MSOinHOL-deep-subst-lemma*
imports *MSOinHOL-deep*
begin

4.1 First-order machinery (Part A)

Free and bound first-order variable occurrences.

primrec *is-free* (**infix** *free'-in* 900)

where

$x \text{ free-in } (r^d(u,v)) = (x = u \vee x = v)$
 $| x \text{ free-in } (X^d(z)) = (x = z)$
 $| x \text{ free-in } (\neg^d \varphi) = (x \text{ free-in } \varphi)$
 $| x \text{ free-in } (\varphi \wedge^d \psi) = (x \text{ free-in } \varphi \vee x \text{ free-in } \psi)$
 $| x \text{ free-in } (\exists^d y. \varphi) = (x \text{ free-in } \varphi \wedge x \neq y)$
 $| x \text{ free-in } (\exists^d_2 Y. \varphi) = (x \text{ free-in } \varphi)$

abbreviation *is-not-free* (**infix** *not'-free'-in* 900)

where $x \text{ not-free-in } \varphi \equiv \neg (x \text{ free-in } \varphi)$

fun *is-bound* (**infix** *bound'-in* 900)

where

$x \text{ bound-in } (r^d(u,v)) = \text{False}$
 $| x \text{ bound-in } (X^d(z)) = \text{False}$
 $| x \text{ bound-in } (\neg^d \varphi) = (x \text{ bound-in } \varphi)$
 $| x \text{ bound-in } (\varphi \wedge^d \psi) = (x \text{ bound-in } \varphi \vee x \text{ bound-in } \psi)$
 $| x \text{ bound-in } (\exists^d y. \varphi) = (x = y \vee x \text{ bound-in } \varphi)$
 $| x \text{ bound-in } (\exists^d_2 Y. \varphi) = (x \text{ bound-in } \varphi)$

abbreviation *is-not-bound* (**infix** *not'-bound'-in* 900)

where $x \text{ not-bound-in } \varphi \equiv \neg (x \text{ bound-in } \varphi)$

abbreviation *occurs* (**infix** *occurs'-in* 900)

where $x \text{ occurs-in } \varphi \equiv x \text{ free-in } \varphi \vee x \text{ bound-in } \varphi$

abbreviation *not-in* (**infix** *not'-in* 900)

where $x \text{ not-in } \varphi \equiv x \text{ not-free-in } \varphi \wedge x \text{ not-bound-in } \varphi$

A fresh first-order variable: strictly larger than every first-order variable occurring in φ .

primrec *fresh* (*fresh'(-')*)

where

$\text{fresh } (r^d(u,v)) = \max (u+1) (v+1)$
 $| \text{fresh } (X^d(z)) = z+1$
 $| \text{fresh } (\neg^d \varphi) = \text{fresh } \varphi$
 $| \text{fresh } (\varphi \wedge^d \psi) = \max (\text{fresh } \varphi) (\text{fresh } \psi)$
 $| \text{fresh } (\exists^d x. \varphi) = \max (x+1) (\text{fresh } \varphi)$
 $| \text{fresh } (\exists^d_2 Y. \varphi) = \text{fresh } \varphi$

lemma *L5*: $x \text{ bound-in } \varphi \implies x < (\text{fresh } \varphi)$
by (*induct* φ) *auto*

lemma *L6*: $x \text{ free-in } \varphi \implies x < (\text{fresh } \varphi)$
by (*induct* φ) *auto*

lemma *L7*: $(\text{fresh } \varphi) \text{ not-in } \varphi$
using *L5 L6* **by** *blast*

lemma *L8*: $\text{max } (\text{fresh } \varphi) (\text{fresh } \psi) \text{ not-free-in } \varphi$
by (*metis* *L6 L7 max.absorb3 max-def*)

lemma *L9*: $\text{max } (\text{fresh } \varphi) (\text{fresh } \psi) \text{ not-bound-in } \varphi$
by (*metis* *L5 L7 max.absorb3 max-def*)

lemma *L10*: $\text{max } (\text{fresh } \varphi) (\text{fresh } \psi) \text{ not-free-in } \psi$
by (*metis* *L8 max commute*)

lemma *L11*: $\text{max } (\text{fresh } \varphi) (\text{fresh } \psi) \text{ not-bound-in } \psi$
by (*metis* *L9 max commute*)

Irrelevance lemma: updating a non-free first-order variable does not affect truth.

lemma *L12*:
 $y \text{ not-free-in } \varphi \implies (\langle I, D, E \rangle, g[y \leftarrow d], G \models^d \varphi) = (\langle I, D, E \rangle, g, G \models^d \varphi)$
by (*induct* φ *arbitrary*: $g \ G$; *simp*; *metis* *L4 L2*)

First-order variable-for-variable substitution. The second-order binder descends transparently.

primrec *Subst* ($[\leftarrow]'$)
where
 $[x \leftarrow z](r^d(u, v)) = r^d(\text{if } x = u \text{ then } z \text{ else } u, \text{if } x = v \text{ then } z \text{ else } v)$
 $| [x \leftarrow z](X^d(w)) = X^d(\text{if } x = w \text{ then } z \text{ else } w)$
 $| [x \leftarrow z](\neg^d \varphi) = \neg^d([x \leftarrow z](\varphi))$
 $| [x \leftarrow z](\varphi \wedge^d \psi) = ([x \leftarrow z](\varphi) \wedge^d [x \leftarrow z](\psi))$
 $| [x \leftarrow z](\exists^d y. \varphi) = (\text{if } x = y \text{ then } (\exists^d y. \varphi) \text{ else } (\exists^d y. [x \leftarrow z](\varphi)))$
 $| [x \leftarrow z](\exists^d_2 Y. \varphi) = (\exists^d_2 Y. [x \leftarrow z](\varphi))$

lemma *L13* [*simp*]: $\text{size } [x \leftarrow z](\varphi) = \text{size } \varphi$
by (*induct* φ ; *auto*)

lemma *L14* [*simp*]: $[x \leftarrow x](\varphi) = \varphi$
by (*induct* φ ; *auto*)

lemma *L15*:
assumes $x \neq a$
shows $[a \leftarrow z]([a \leftarrow x](\varphi)) = [a \leftarrow x](\varphi)$
using *assms* **by** (*induct* φ) *auto*

lemma *L16* [*simp*]:

assumes $a \neq x$
shows $a \text{ not-free-in } ([a \leftarrow x](\varphi))$
using *assms* **by** (*induct* φ) *auto*

Size-based induction principle (size-based on both existential binders).

lemma *SInduct*:

assumes $\bigwedge r \ u \ v. P \ (r^d(u,v))$
and $\bigwedge X \ z. P \ (X^d(z))$
and $\bigwedge \varphi. (\bigwedge \psi. \text{size } \psi \leq \text{size } \varphi \implies P \ \psi) \implies P \ (\neg^d \varphi)$
and $\bigwedge \varphi \ \psi. (\bigwedge \chi. \text{size } \chi \leq \text{size } \varphi + \text{size } \psi \implies P \ \chi) \implies P \ (\varphi \wedge^d \psi)$
and $\bigwedge y \ \varphi. (\bigwedge \psi. \text{size } \psi \leq \text{size } \varphi \implies P \ \psi) \implies P \ (\exists^d y. \varphi)$
and $\bigwedge Y \ \varphi. (\bigwedge \psi. \text{size } \psi \leq \text{size } \varphi \implies P \ \psi) \implies P \ (\exists^d_2 Y. \varphi)$
shows $P \ \varphi$
using *assms*
proof (*induct size* φ *arbitrary:* φ *rule: less-induct*)
case less **thus** ?*case* **by** (*induct* φ) *auto*
qed

Stronger induction: structural for the propositional cases, size-based for the two binders.

lemma *QInduct*:

assumes $\bigwedge r \ u \ v. P \ (r^d(u,v))$
and $\bigwedge X \ z. P \ (X^d(z))$
and $\bigwedge \varphi. P \ \varphi \implies P \ (\neg^d \varphi)$
and $\bigwedge \varphi \ \psi. P \ \varphi \implies P \ \psi \implies P \ (\varphi \wedge^d \psi)$
and $\bigwedge y \ \varphi. (\bigwedge \psi. \text{size } \psi \leq \text{size } \varphi \implies P \ \psi) \implies P \ (\exists^d y. \varphi)$
and $\bigwedge Y \ \varphi. (\bigwedge \psi. \text{size } \psi \leq \text{size } \varphi \implies P \ \psi) \implies P \ (\exists^d_2 Y. \varphi)$
shows $P \ \varphi$
using *assms* **by** (*induct* φ *rule: SInduct*) *auto*

Substitutability predicate: z may safely replace x in φ without capture.

primrec *SubstitutableForIn* (*- is'-subst'-for - in -* [*999,1,999*] *999*)

where

$z \text{ is-subst-for } x \text{ in } (r^d(u,v)) = \text{True}$
 $| z \text{ is-subst-for } x \text{ in } (X^d(w)) = \text{True}$
 $| z \text{ is-subst-for } x \text{ in } (\neg^d \varphi) = (z \text{ is-subst-for } x \text{ in } \varphi)$
 $| z \text{ is-subst-for } x \text{ in } (\varphi \wedge^d \psi) = (z \text{ is-subst-for } x \text{ in } \varphi \wedge z \text{ is-subst-for } x \text{ in } \psi)$
 $| z \text{ is-subst-for } x \text{ in } (\exists^d y. \varphi) = (y = x \vee (x \text{ not-free-in } \varphi \vee y \neq x) \wedge z \text{ is-subst-for } x \text{ in } \varphi)$
 $| z \text{ is-subst-for } x \text{ in } (\exists^d_2 Y. \varphi) = (z \text{ is-subst-for } x \text{ in } \varphi)$

Substitution lemma: a syntactic z -for- x substitution corresponds to updating the assignment.

lemma *SubstitutionLemma* [*simp*]:

assumes $z \text{ is-subst-for } x \text{ in } \varphi$
shows $(\langle I, D, E \rangle, g, G \models^d ([x \leftarrow z](\varphi))) = (\langle I, D, E \rangle, g[x \leftarrow (g \ z)], G \models^d \varphi)$
using *assms* **by** (*induction* φ *arbitrary:* $g \ G$; *auto simp: L12 L2*)

Alphabetic renaming preparing capture-avoiding substitution.

fun *ren-for-subst*

where

ren-for-subst $x\ z\ (r^d(u,v)) = r^d(u,v)$
 $| \text{ren-for-subst } x\ z\ (X^d(w)) = X^d(w)$
 $| \text{ren-for-subst } x\ z\ (\neg^d \varphi) = \neg^d(\text{ren-for-subst } x\ z\ \varphi)$
 $| \text{ren-for-subst } x\ z\ (\varphi \wedge^d \psi) = (\text{ren-for-subst } x\ z\ \varphi \wedge^d \text{ren-for-subst } x\ z\ \psi)$
 $| \text{ren-for-subst } x\ z\ (\exists^d y. \varphi) =$
 (*if* $y = z \wedge x \text{ free-in } \varphi$
 then $\text{let } f = \max(\text{fresh } \varphi)\ (z+1); \varphi' = [y \leftarrow f](\varphi)$
 in $(\exists^d f. \text{ren-for-subst } x\ z\ \varphi')$
 else $\exists^d y. \text{ren-for-subst } x\ z\ \varphi)$
 $| \text{ren-for-subst } x\ z\ (\exists^d_2 Y. \varphi) = \exists^d_2 Y. \text{ren-for-subst } x\ z\ \varphi$

lemma *L17* [*simp*]: $\text{size}(\text{ren-for-subst } x\ z\ \varphi) = \text{size } \varphi$

by (*induct* φ *arbitrary*; $z\ x$ *rule*: *QInduct*; *simp* *add*: *Let-def*)

lemma *L18*: $\alpha \text{ not-in } \varphi \implies \alpha \text{ is-subst-for } \beta \text{ in } \varphi$

by (*induct* φ) *auto*

lemma *L19*: $x \text{ free-in } \psi \implies y \neq x \implies x \text{ free-in } [y \leftarrow z](\psi)$

by (*induct* ψ) *auto*

lemma *L20* [*simp*]:

$x \text{ free-in } \text{ren-for-subst } x\ z\ \varphi = (x \text{ free-in } \varphi)$

by (*induct* φ *rule*: *QInduct*; *simp* *add*: *Let-def*)

(*metis* *L16 L6 L7 L19 max.absorb3 max-def-raw*)

lemma *L21*:

$(\langle I, D, E \rangle, g, G \models^d \varphi) = (\langle I, D, E \rangle, g, G \models^d (\text{ren-for-subst } x\ z\ \varphi))$

by (*induct* φ *arbitrary*; $z\ g\ G$ *rule*: *QInduct*;

simp *add*: *Let-def*;

smt (*verit*) *L12 L18 L2 L3 L5 L6 SubstitutionLemma*

max.strict-order-iff max-def)

lemma *L22*: $z \text{ is-subst-for } x \text{ in } (\text{ren-for-subst } x\ z\ \varphi)$

by (*induct* φ *rule*: *QInduct*; *simp*;

metis *L13 SubstitutableForIn.simps(5)*

Suc-n-not-le-n dual-order.refl max.cobounded2)

lemma *L23*: $x \text{ is-subst-for } x \text{ in } \varphi$

by (*induct* φ) *auto*

lemma *L24*: $x \text{ not-free-in } \varphi \implies [x \leftarrow z](\varphi) = \varphi$

by (*induct* φ) *auto*

lemma *L26* [*simp*]: $z \text{ bound-in } [x \leftarrow y](\varphi) = z \text{ bound-in } \varphi$

by (*induct* φ) *auto*

Safe (capture-avoiding) first-order substitution: rename first, then substitute.

definition *ren-subst* ($[- \leftarrow_r -]'(-')$)
where $[x \leftarrow_r z](\varphi) = [x \leftarrow z](\text{ren-for-subst } x \ z \ \varphi)$

lemma *L27 [simp]*:
 $(\langle I, D, E \rangle, g, G \models^d [x \leftarrow_r z](\varphi)) = (\langle I, D, E \rangle, g[x \leftarrow g \ z], G \models^d \varphi)$
using *L21 L22 ren-subst-def by auto*

lemma *L28 [simp]*: $\text{size } ([x \leftarrow_r z](\varphi)) = \text{size } \varphi$
by (*induct* φ *rule: QInduct*) (*auto simp: ren-subst-def Let-def*)

lemma *L29*:
 $g \text{ onto } D \implies (\langle I, D, E \rangle, g, G \models^d (\exists^d x. \varphi)) = (\exists z. \langle I, D, E \rangle, g, G \models^d ([x \leftarrow_r z](\varphi)))$
by (*induct* φ *arbitrary: I g G x rule: QInduct; simp; blast*)

4.2 Second-order machinery (Part B)

primrec *is-free2* (**infix** *free2'-in* 900)

where
 $X \text{ free2-in } (r^d(u, v)) = \text{False}$
 $| X \text{ free2-in } (Y^d(z)) = (X = Y)$
 $| X \text{ free2-in } (\neg^d \varphi) = (X \text{ free2-in } \varphi)$
 $| X \text{ free2-in } (\varphi \wedge^d \psi) = (X \text{ free2-in } \varphi \vee X \text{ free2-in } \psi)$
 $| X \text{ free2-in } (\exists^d y. \varphi) = (X \text{ free2-in } \varphi)$
 $| X \text{ free2-in } (\exists^d_2 Y. \varphi) = (X \text{ free2-in } \varphi \wedge X \neq Y)$

abbreviation *is-not-free2* (**infix** *not'-free2'-in* 900)
where $X \text{ not-free2-in } \varphi \equiv \neg (X \text{ free2-in } \varphi)$

fun *is-bound2* (**infix** *bound2'-in* 900)

where
 $X \text{ bound2-in } (r^d(u, v)) = \text{False}$
 $| X \text{ bound2-in } (Y^d(z)) = \text{False}$
 $| X \text{ bound2-in } (\neg^d \varphi) = (X \text{ bound2-in } \varphi)$
 $| X \text{ bound2-in } (\varphi \wedge^d \psi) = (X \text{ bound2-in } \varphi \vee X \text{ bound2-in } \psi)$
 $| X \text{ bound2-in } (\exists^d y. \varphi) = (X \text{ bound2-in } \varphi)$
 $| X \text{ bound2-in } (\exists^d_2 Y. \varphi) = (X = Y \vee X \text{ bound2-in } \varphi)$

abbreviation *is-not-bound2* (**infix** *not'-bound2'-in* 900)
where $X \text{ not-bound2-in } \varphi \equiv \neg (X \text{ bound2-in } \varphi)$

abbreviation *not-in2* (**infix** *not2'-in* 900)

where $X \text{ not2-in } \varphi \equiv X \text{ not-free2-in } \varphi \wedge X \text{ not-bound2-in } \varphi$

primrec *fresh2*

where
 $\text{fresh2 } (r^d(u, v)) = 0$
 $| \text{fresh2 } (Y^d(z)) = Y + 1$

| $\text{fresh2 } (\neg^d \varphi) = \text{fresh2 } \varphi$
| $\text{fresh2 } (\varphi \wedge^d \psi) = \max (\text{fresh2 } \varphi) (\text{fresh2 } \psi)$
| $\text{fresh2 } (\exists^d y. \varphi) = \text{fresh2 } \varphi$
| $\text{fresh2 } (\exists^d_2 Y. \varphi) = \max (Y+1) (\text{fresh2 } \varphi)$

lemma *N5*: $X \text{ bound2-in } \varphi \implies X < (\text{fresh2 } \varphi)$
by (*induct* φ) *auto*

lemma *N6*: $X \text{ free2-in } \varphi \implies X < (\text{fresh2 } \varphi)$
by (*induct* φ) *auto*

lemma *N7*: $(\text{fresh2 } \varphi) \text{ not2-in } \varphi$
using *N5 N6* **by** *blast*

lemma *N8*: $\max (\text{fresh2 } \varphi) (\text{fresh2 } \psi) \text{ not-free2-in } \varphi$
by (*metis* *N6 N7 max.absorb3 max-def*)

lemma *N9*: $\max (\text{fresh2 } \varphi) (\text{fresh2 } \psi) \text{ not-bound2-in } \varphi$
by (*metis* *N5 N7 max.absorb3 max-def*)

lemma *N10*: $\max (\text{fresh2 } \varphi) (\text{fresh2 } \psi) \text{ not-free2-in } \psi$
by (*metis* *N8 max commute*)

lemma *N11*: $\max (\text{fresh2 } \varphi) (\text{fresh2 } \psi) \text{ not-bound2-in } \psi$
by (*metis* *N9 max commute*)

Irrelevance lemma for second-order assignments.

lemma *N12*:
 $Y \text{ not-free2-in } \varphi \implies (\langle I, D, E \rangle, g, G \langle Y \leftarrow S \rangle \models^d \varphi) = (\langle I, D, E \rangle, g, G \models^d \varphi)$
by (*induct* φ *arbitrary*: $g \ G$; *simp*; *metis* *M4 M2*)

Second-order variable-for-variable substitution. The first-order binder descends transparently.

primrec *Subst2* ($[\leftarrow_2]$)'($\cdot$)
where
 $[X \leftarrow_2 Z](r^d(u, v)) = r^d(u, v)$
 $[X \leftarrow_2 Z](Y^d(w)) = (\text{if } X = Y \text{ then } Z \text{ else } Y)^d(w)$
 $[X \leftarrow_2 Z](\neg^d \varphi) = \neg^d([X \leftarrow_2 Z](\varphi))$
 $[X \leftarrow_2 Z](\varphi \wedge^d \psi) = ([X \leftarrow_2 Z](\varphi) \wedge^d [X \leftarrow_2 Z](\psi))$
 $[X \leftarrow_2 Z](\exists^d y. \varphi) = (\exists^d y. [X \leftarrow_2 Z](\varphi))$
 $[X \leftarrow_2 Z](\exists^d_2 Y. \varphi) = (\text{if } X = Y \text{ then } (\exists^d_2 Y. \varphi) \text{ else } (\exists^d_2 Y. [X \leftarrow_2 Z](\varphi)))$

lemma *N13* [*simp*]: $\text{size } [X \leftarrow_2 Z](\varphi) = \text{size } \varphi$
by (*induct* φ ; *auto*)

lemma *N14* [*simp*]: $[X \leftarrow_2 X](\varphi) = \varphi$
by (*induct* φ ; *auto*)

lemma *N16* [*simp*]:

assumes $A \neq X$
 shows A not-free2-in $([A \leftarrow_2 X](\varphi))$
 using assms by (induct φ) auto

primrec *SubstitutableForIn2* (- is'-subst2'-for - in - [999,1,999] 999)

where

Z is-subst2-for X in $(r^d(u,v)) = \text{True}$
 $|$ Z is-subst2-for X in $(Y^d(w)) = \text{True}$
 $|$ Z is-subst2-for X in $(\neg^d \varphi) = (Z \text{ is-subst2-for } X \text{ in } \varphi)$
 $|$ Z is-subst2-for X in $(\varphi \wedge^d \psi) = (Z \text{ is-subst2-for } X \text{ in } \varphi \wedge Z \text{ is-subst2-for } X \text{ in } \psi)$
 $|$ Z is-subst2-for X in $(\exists^d y. \varphi) = (Z \text{ is-subst2-for } X \text{ in } \varphi)$
 $|$ Z is-subst2-for X in $(\exists^d_2 Y. \varphi) = (Y = X \vee (X \text{ not-free2-in } \varphi \vee Y \neq Z) \wedge Z \text{ is-subst2-for } X \text{ in } \varphi)$

lemma *SubstitutionLemma2* [simp]:

assumes Z is-subst2-for X in φ

shows $(\langle I, D, E \rangle, g, G \models^d ([X \leftarrow_2 Z](\varphi))) = (\langle I, D, E \rangle, g, G \langle X \leftarrow (G Z) \rangle \models^d \varphi)$

using assms by (induction φ arbitrary: $g G$; auto simp: N12 M2)

fun *ren-for-subst2*

where

$\text{ren-for-subst2 } X Z (r^d(u,v)) = r^d(u,v)$
 $|$ $\text{ren-for-subst2 } X Z (Y^d(w)) = Y^d(w)$
 $|$ $\text{ren-for-subst2 } X Z (\neg^d \varphi) = \neg^d (\text{ren-for-subst2 } X Z \varphi)$
 $|$ $\text{ren-for-subst2 } X Z (\varphi \wedge^d \psi) = (\text{ren-for-subst2 } X Z \varphi \wedge^d \text{ren-for-subst2 } X Z \psi)$
 $|$ $\text{ren-for-subst2 } X Z (\exists^d y. \varphi) = \exists^d y. \text{ren-for-subst2 } X Z \varphi$
 $|$ $\text{ren-for-subst2 } X Z (\exists^d_2 Y. \varphi) =$
 (if $Y = Z \wedge X \text{ free2-in } \varphi$
 then let $f = \max (\text{fresh2 } \varphi) (Z+1)$; $\varphi' = [Y \leftarrow_2 f](\varphi)$
 in $(\exists^d_2 f. \text{ren-for-subst2 } X Z \varphi')$
 else $\exists^d_2 Y. \text{ren-for-subst2 } X Z \varphi)$

lemma N17 [simp]: $\text{size } (\text{ren-for-subst2 } X Z \varphi) = \text{size } \varphi$

by (induct φ arbitrary: $Z X$ rule: QInduct; simp add: Let-def)

lemma N18: $\alpha \text{ not2-in } \varphi \implies \alpha \text{ is-subst2-for } \beta \text{ in } \varphi$

by (induct φ) auto

lemma N19: $X \text{ free2-in } \psi \implies Y \neq X \implies X \text{ free2-in } [Y \leftarrow_2 Z](\psi)$

by (induct ψ) auto

lemma N20 [simp]:

$X \text{ free2-in } \text{ren-for-subst2 } X Z \varphi = (X \text{ free2-in } \varphi)$

by (induct φ rule: QInduct; simp add: Let-def)

(metis N16 N6 N7 N19 max.absorb3 max-def-raw)

lemma N21:

$(\langle I, D, E \rangle, g, G \models^d \varphi) = (\langle I, D, E \rangle, g, G \models^d (\text{ren-for-subst2 } X Z \varphi))$

by (*induct* φ *arbitrary*: $Z \ g \ G$ *rule*: $QInduct$;
simp add: $Let\text{-}def$;
smt (*verit*) $N12 \ N18 \ M2 \ M3 \ N5 \ N6 \ SubstitutionLemma2$
 $max.strict\text{-}order\text{-}iff \ max\text{-}def$)

lemma $N22$: Z *is-subst2-for* X *in* (*ren-for-subst2* $X \ Z \ \varphi$)
by (*induct* φ *rule*: $QInduct$; *simp*;
metis $N13 \ SubstitutableForIn2.simps(6)$
 $Suc\text{-}n\text{-}not\text{-}le\text{-}n \ dual\text{-}order.refl \ max.cobounded2$)

definition *ren-subst2* ($[- \leftarrow_{r2} -]'(-')$)
where $[X \leftarrow_{r2} Z](\varphi) = [X \leftarrow_2 Z](ren\text{-}for\text{-}subst2 \ X \ Z \ \varphi)$

lemma $N27$ [*simp*]:
 $(\langle I, D, E \rangle, g, G \models^d [X \leftarrow_{r2} Z](\varphi)) = (\langle I, D, E \rangle, g, G \langle X \leftarrow G \ Z \rangle \models^d \varphi)$
using $N21 \ N22 \ ren\text{-}subst2\text{-}def$ **by** *auto*

lemma $N28$ [*simp*]: $size \ ([X \leftarrow_{r2} Z](\varphi)) = size \ \varphi$
by (*induct* φ *rule*: $QInduct$) (*auto simp*: $ren\text{-}subst2\text{-}def \ Let\text{-}def$)

lemma $N29$:
 $G \text{ onto } E \implies (\langle I, D, E \rangle, g, G \models^d (\exists^d_2 X. \varphi)) = (\exists Z. \langle I, D, E \rangle, g, G \models^d ([X \leftarrow_{r2} Z](\varphi)))$
by (*induct* φ *arbitrary*: $I \ g \ G \ X$ *rule*: $QInduct$; *simp*; *blast*)

end

4.3 Auxiliary renaming results

theory $MSOinHOL\text{-}subst\text{-}extras$
imports $MSOinHOL\text{-}deep\text{-}subst\text{-}lemma$
begin

Explicit rename-evaluation lemmas: renaming a bound variable to a fresh f and updating the assignment preserves truth—the semantic core behind $L21$ / $N21$.

First-order: rename y to a fresh f , then evaluate.

lemma *rename-eval*:
assumes *fresh* $\varphi \leq f$ **and** $y < f$
shows $(\langle I, D, E \rangle, g[f \leftarrow d], G \models^d [y \leftarrow f](\varphi)) = (\langle I, D, E \rangle, g[y \leftarrow d], G \models^d \varphi)$
proof –
let $?g' = g[f \leftarrow d]$
have $fy: f \neq y$ **using** $assms(2)$ **by** *simp*
have $nf: f \text{ not-free-in } \varphi$ **using** $assms(1)$ **by** (*meson* $L6 \ leD$)
have $f \text{ not-in } \varphi$ **using** $assms(1)$ **by** (*meson* $L5 \ L6 \ leD$)
hence *sub*: f *is-subst-for* y *in* φ **by** (*rule* $L18$)
have *swap*: $?g'[y \leftarrow d] = (g[y \leftarrow d])[f \leftarrow d]$
using fy **by** (*rule* $L2$)
have $(\langle I, D, E \rangle, ?g', G \models^d [y \leftarrow f](\varphi)) = (\langle I, D, E \rangle, ?g'[y \leftarrow (?g' f)], G \models^d \varphi)$

```

    using sub by (rule SubstitutionLemma)
    also have ... = ( $\langle I, D, E \rangle, ?g[y \leftarrow d], G \models^d \varphi$ ) by simp
    also have ... = ( $\langle I, D, E \rangle, (g[y \leftarrow d])[f \leftarrow d], G \models^d \varphi$ )
    using swap by simp
    also have ... = ( $\langle I, D, E \rangle, g[y \leftarrow d], G \models^d \varphi$ )
    using nf by (simp add: L12)
    finally show ?thesis .
qed

```

Second-order: rename Y to a fresh f , then evaluate. (The monadic-set twin of *rename-eval*.)

```

lemma rename-eval2:
  assumes fresh2  $\varphi \leq f$  and  $Y < f$ 
  shows ( $\langle I, D, E \rangle, g, G \langle f \leftarrow S \rangle \models^d [Y \leftarrow_2 f](\varphi)$ ) = ( $\langle I, D, E \rangle, g, G \langle Y \leftarrow S \rangle \models^d \varphi$ )
proof -
  let ?G' =  $G \langle f \leftarrow S \rangle$ 
  have fy:  $f \neq Y$  using assms(2) by simp
  have nf:  $f$  not-free2-in  $\varphi$  using assms(1) by (meson N6 leD)
  have f not2-in  $\varphi$  using assms(1) by (meson N5 N6 leD)
  hence sub:  $f$  is-subst2-for  $Y$  in  $\varphi$  by (rule N18)
  have swap:  $?G' \langle Y \leftarrow S \rangle = G \langle Y \leftarrow S \rangle \langle f \leftarrow S \rangle$ 
  using fy by (rule M2)
  have ( $\langle I, D, E \rangle, g, ?G' \models^d [Y \leftarrow_2 f](\varphi)$ ) = ( $\langle I, D, E \rangle, g, ?G' \langle Y \leftarrow (?G' f) \rangle \models^d \varphi$ )
  using sub by (rule SubstitutionLemma2)
  also have ... = ( $\langle I, D, E \rangle, g, ?G' \langle Y \leftarrow S \rangle \models^d \varphi$ ) by simp
  also have ... = ( $\langle I, D, E \rangle, g, (G \langle Y \leftarrow S \rangle) \langle f \leftarrow S \rangle \models^d \varphi$ )
  using swap by simp
  also have ... = ( $\langle I, D, E \rangle, g, G \langle Y \leftarrow S \rangle \models^d \varphi$ )
  using nf by (simp add: N12)
  finally show ?thesis .
qed
end

```

5 Shallow embeddings

5.1 Maximal-shallow embedding

```

theory MSOinHOL-shallow
  imports MSOinHOL-preliminaries
begin

```

Maximal (heavyweight) shallow embedding of MSO in HOL; MSO formulas are HOL terms of the following type σ .

```

type-synonym  $\sigma = \mathcal{I} \Rightarrow \mathcal{D} \Rightarrow \mathcal{P} \Rightarrow \mathcal{E} \Rightarrow \mathcal{G} \Rightarrow bool$ 

```

The six primitive cases.

```

definition AtmS ::  $R \Rightarrow V \Rightarrow V \Rightarrow \sigma$  ( $-^s'(-, -)$ )

```

where $r^s(x,y) \equiv \lambda I D E g G. I r (g x) (g y)$

definition $PrdS :: V2 \Rightarrow V \Rightarrow \sigma \ (-^s'(-'))$
where $X^s(x) \equiv \lambda I D E g G. (G X) (g x)$

definition $NegS :: \sigma \Rightarrow \sigma \ (-^s - [58] \ 59)$
where $\neg^s \varphi \equiv \lambda I D E g G. \neg (\varphi I D E g G)$

definition $AndS :: \sigma \Rightarrow \sigma \Rightarrow \sigma \ (\mathbf{infixr} \wedge^s \ 56)$
where $\varphi \wedge^s \psi \equiv \lambda I D E g G. \varphi I D E g G \wedge \psi I D E g G$

definition $ExS :: V \Rightarrow \sigma \Rightarrow \sigma \ (\exists^s -. - \ 53)$
where $\exists^s x. \varphi \equiv \lambda I D E g G. \exists d:D. \varphi I D E (g[x \leftarrow d]) G$

definition $ExS2 :: V2 \Rightarrow \sigma \Rightarrow \sigma \ (\exists^s_2 -. - \ 53)$
where $\exists^s_2 X. \varphi \equiv \lambda I D E g G. \exists S:E. \varphi I D E g (G \langle X \leftarrow S \rangle)$

Derived connectives.

definition $OrS :: \sigma \Rightarrow \sigma \Rightarrow \sigma \ (\mathbf{infixr} \vee^s \ 54)$
where $\varphi \vee^s \psi \equiv \neg^s (\neg^s \varphi \wedge^s \neg^s \psi)$

definition $ImpS :: \sigma \Rightarrow \sigma \Rightarrow \sigma \ (\mathbf{infixr} \supset^s \ 55)$
where $\varphi \supset^s \psi \equiv \neg^s \varphi \vee^s \psi$

definition $IffS :: \sigma \Rightarrow \sigma \Rightarrow \sigma \ (\mathbf{infixr} \longleftrightarrow^s \ 54)$
where $\varphi \longleftrightarrow^s \psi \equiv (\varphi \supset^s \psi) \wedge^s (\psi \supset^s \varphi)$

definition $AllS :: V \Rightarrow \sigma \Rightarrow \sigma \ (\forall^s -. - \ 53)$
where $\forall^s x. \varphi \equiv \neg^s (\exists^s x. \neg^s \varphi)$

definition $AllS2 :: V2 \Rightarrow \sigma \Rightarrow \sigma \ (\forall^s_2 -. - \ 53)$
where $\forall^s_2 X. \varphi \equiv \neg^s (\exists^s_2 X. \neg^s \varphi)$

Relative truth and validity (mirroring the deep embedding).

definition $RelTruthS :: \mathcal{I} \Rightarrow \mathcal{D} \Rightarrow \mathcal{P} \Rightarrow \mathcal{E} \Rightarrow \mathcal{G} \Rightarrow \sigma \Rightarrow bool$
 $(\langle \neg, -, \cdot \rangle, -, - \models^s - [100, 0, 0, 0, 0] \ 100)$
where $\langle I, D, E \rangle, g, G \models^s \varphi \equiv \varphi I D E g G$

definition $ValS :: \sigma \Rightarrow bool \ (\models^s - \ 9)$
where $\models^s \varphi \equiv \forall I D E g G. g \text{ into } D \longrightarrow G \text{ into } E \longrightarrow \langle I, D, E \rangle, g, G \models^s \varphi$

Auxiliary “full-domain” notion of validity: assignments range over the full types.

definition $ValS' (\models^{s''} - \ 9)$
where $\models^{s'} \varphi \equiv \forall I g G. \langle I, Univ, Univ \rangle, g, G \models^s \varphi$

General validity implies full-domain validity.

lemma $Val\text{-}s: \models^s \varphi \implies \models^{s'} \varphi$
using $ValS'\text{-}def \ ValS\text{-}def$ **by** *simp*

Bag of definitions.

```

named-theorems DefS
lemmas DefS-defs [DefS] =
  AtmS-def PrdS-def NegS-def AndS-def ExS-def ExS2-def
  OrS-def ImpS-def IffS-def AllS-def AllS2-def
  RelTruthS-def ValS-def ValS'-def

end

```

5.2 Minimal-shallow embedding (locale)

```

theory MSOinHOL-shallow-minimal-locale
imports MSOinHOL-preliminaries
begin

```

Minimal (lightweight) shallow embedding of MSO in HOL, packaged as a locale. Since MSO carries no world dependency, the formula type collapses to *bool*.

```

locale MinS =
  fixes II ::  $\mathcal{I}$  and gg ::  $\mathcal{E}$  and GG ::  $\mathcal{G}$ 
begin

```

Six primitive cases. *ExM*, *ExM2* are HOL binders over the symbol types *V*, *V2*; atoms consult *II* via *gg*, and membership consults *GG* via *gg*.

```

definition AtmM ::  $R \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$  ( $\neg^m(-, -')$ )
  where  $r^m(x, y) \equiv II\ r\ (gg\ x)\ (gg\ y)$ 

```

```

definition PrdM ::  $V2 \Rightarrow V \Rightarrow \text{bool}$  ( $\neg^m(-, -')$ )
  where  $X^m(x) \equiv (GG\ X)\ (gg\ x)$ 

```

```

definition NegM ::  $\text{bool} \Rightarrow \text{bool}$  ( $\neg^m - [58]\ 59$ )
  where  $\neg^m\varphi \equiv \neg\varphi$ 

```

```

definition AndM ::  $\text{bool} \Rightarrow \text{bool} \Rightarrow \text{bool}$  (infixr  $\wedge^m$  56)
  where  $\varphi \wedge^m \psi \equiv \varphi \wedge \psi$ 

```

```

definition ExM ::  $(V \Rightarrow \text{bool}) \Rightarrow \text{bool}$  (binder  $\exists^m$  53)
  where  $\exists^m d. \Phi\ d \equiv \exists d. \Phi\ d$ 

```

```

definition ExM2 ::  $(V2 \Rightarrow \text{bool}) \Rightarrow \text{bool}$  (binder  $\exists^m_2$  53)
  where  $\exists^m_2 D. \Phi\ D \equiv \exists D. \Phi\ D$ 

```

Derived connectives.

```

definition OrM ::  $\text{bool} \Rightarrow \text{bool} \Rightarrow \text{bool}$  (infixr  $\vee^m$  54)
  where  $\varphi \vee^m \psi \equiv \neg^m(\neg^m\varphi \wedge^m \neg^m\psi)$ 

```

```

definition ImpM ::  $\text{bool} \Rightarrow \text{bool} \Rightarrow \text{bool}$  (infixr  $\supset^m$  55)
  where  $\varphi \supset^m \psi \equiv \neg^m\varphi \vee^m \psi$ 

```

definition *IffM* :: *bool* $\Rightarrow$ *bool* $\Rightarrow$ *bool* (**infixr** $\longleftrightarrow^m$ 54)

where $\varphi \longleftrightarrow^m \psi \equiv (\varphi \supset^m \psi) \wedge^m (\psi \supset^m \varphi)$

definition *AllM* :: (*V* $\Rightarrow$ *bool*) $\Rightarrow$ *bool* (**binder** $\forall^m$ 53)

where $\forall^m d. \Phi \ d \equiv \forall d. \Phi \ d$

definition *AllM2* :: (*V2* $\Rightarrow$ *bool*) $\Rightarrow$ *bool* (**binder** $\forall^{m_2}$ 53)

where $\forall^{m_2} D. \Phi \ D \equiv \forall D. \Phi \ D$

Relative truth and validity. As the formula type is *bool*, validity is the identity.

definition *ValM* :: *bool* $\Rightarrow$ *bool* ($\models^m$ - 9)

where $\models^m \varphi \equiv \varphi$

Bag of definitions.

named-theorems *DefM*

lemmas *DefM-defs* [*DefM*] =

AtmM-def PrdM-def NegM-def AndM-def ExM-def ExM2-def
OrM-def ImpM-def IffM-def AllM-def AllM2-def ValM-def

end

end

5.3 Minimal-shallow embedding (translation)

theory *MSOinHOL-shallow-minimal*

imports *MSOinHOL-faithfulness-locale*

begin

Constants-only presentation: a global interpretation of the minimal shallow embedding.

consts *II* :: $\mathcal{I}$ *gg* :: $\mathcal{E}$ *GG* :: $\mathcal{G}$

global-interpretation *MinS II gg GG* .

Re-introduce the locale-internal notation at the global level.

notation *AtmM* $(\neg^{m'}(-, -))$

and *PrdM* $(\neg^{m'}(-))$

and *NegM* $(\neg^m - [58] \ 59)$

and *AndM* (**infixr** $\wedge^m$ 56)

and *ExM* (**binder** $\exists^m$ 53)

and *ExM2* (**binder** $\exists^{m_2}$ 53)

and *OrM* (**infixr** $\vee^m$ 54)

and *ImpM* (**infixr** $\supset^m$ 55)

and *IffM* (**infixr** $\longleftrightarrow^m$ 54)

and *AllM* (**binder** $\forall^m$ 53)

and *AllM2* (**binder** $\forall^{m_2}$ 53)

```

and ValM ( $\models^m$  - 9)
and DpToShM ( $\llbracket \cdot \rrbracket$ )

end

```

6 Faithfulness

6.1 The faithfulness locale

```

theory MSOinHOL-faithfulness-locale
imports
  MSOinHOL-deep
  MSOinHOL-shallow
  MSOinHOL-shallow-minimal-locale
  MSOinHOL-deep-subst-lemma
begin

```

Translation from deep to maximal shallow (transparent: both use named-variable binders).

```

fun DpToShS ( $\llbracket \cdot \rrbracket$ )
where
  |  $\llbracket r^d(x, y) \rrbracket = r^s(x, y)$ 
  |  $\llbracket X^d(x) \rrbracket = X^s(x)$ 
  |  $\llbracket \neg^d \varphi \rrbracket = \neg^s \llbracket \varphi \rrbracket$ 
  |  $\llbracket \varphi \wedge^d \psi \rrbracket = \llbracket \varphi \rrbracket \wedge^s \llbracket \psi \rrbracket$ 
  |  $\llbracket \exists^d v. \varphi \rrbracket = (\exists^s v. \llbracket \varphi \rrbracket)$ 
  |  $\llbracket \exists^d_2 V. \varphi \rrbracket = (\exists^s_2 V. \llbracket \varphi \rrbracket)$ 

```

Translation from deep to minimal shallow (within the locale *MinS*): each binder is bridged by safe substitution into a HOL-level binder—first-order via $[v \leftarrow_r d]$, second-order via $[V \leftarrow_{r2} D]$.

```

fun (in MinS) DpToShM ( $\llbracket \cdot \rrbracket$ )
where
  |  $\llbracket r^d(x, y) \rrbracket = r^m(x, y)$ 
  |  $\llbracket X^d(x) \rrbracket = X^m(x)$ 
  |  $\llbracket \neg^d \varphi \rrbracket = \neg^m \llbracket \varphi \rrbracket$ 
  |  $\llbracket \varphi \wedge^d \psi \rrbracket = \llbracket \varphi \rrbracket \wedge^m \llbracket \psi \rrbracket$ 
  |  $\llbracket \exists^d v. \varphi \rrbracket = (\exists^m d. \llbracket [v \leftarrow_r d](\varphi) \rrbracket)$ 
  |  $\llbracket \exists^d_2 V. \varphi \rrbracket = (\exists^m_2 D. \llbracket [V \leftarrow_{r2} D](\varphi) \rrbracket)$ 

```

Faithfulness deep $\longleftrightarrow$ maximal shallow. Pointwise first; global by unfolding validity.

theorem *FaithfulSDlem*:

$(\langle I, D, E \rangle, g, G \models^s \llbracket \varphi \rrbracket) \longleftrightarrow (\langle I, D, E \rangle, g, G \models^d \varphi)$
by (*induct* φ *arbitrary*: $g \ G$; *auto simp*: *DefS DefD*)

theorem *FaithfulSD*: $(\models^s \llbracket \varphi \rrbracket) \longleftrightarrow (\models^d \varphi)$

by (*simp add*: *FaithfulSDlem ValD-def ValS-def*)

Faithfulness deep $\longleftrightarrow$ minimal shallow, parametrised by the locale, relative to the ranges of gg and GG .

context *MinS*
begin

theorem *FaithfulMDlem*:

$(\llbracket \varphi \rrbracket) \longleftrightarrow (\langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^d \varphi)$
by (*induct* φ *rule*: *QInduct*; *simp add*: *DefD DefM*) *blast+*

theorem *FaithfulMD*:

$(\models^m \llbracket \varphi \rrbracket) \longleftrightarrow (\langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^d \varphi)$
using *FaithfulMDlem* **by** (*simp add*: *ValM-def*)

Faithfulness minimal shallow $\longleftrightarrow$ maximal shallow, parametrised by the locale, relative to the ranges of gg and GG ; obtained by composing *FaithfulSDlem* and *FaithfulMDlem*.

theorem *FaithfulMSlem*:

$(\llbracket \varphi \rrbracket) \longleftrightarrow (\langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^s \llbracket \varphi \rrbracket)$
using *FaithfulSDlem* *FaithfulMDlem* **by** *auto*

theorem *FaithfulMS*:

$(\models^m \llbracket \varphi \rrbracket) \longleftrightarrow (\langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^s \llbracket \varphi \rrbracket)$
using *FaithfulMSlem* **by** (*simp add*: *ValM-def*)

end

Global faithfulness: a formula holds in every minimal interpretation of *MinS* iff it holds in every model whose first- and second-order domains are exactly the ranges of the chosen assignments.

theorem *FaithfulMS-all*:

$(\forall II \ gg \ GG. \text{MinS.ValM } (\text{MinS.DpToShM } II \ gg \ GG \ \varphi)) = (\forall I \ g \ G. \langle I, \text{Range } g, \text{Range } G \rangle, g, G \models^d \varphi)$
by (*simp add*: *MinS.FaithfulMD*)

One direction toward full deep validity: deep validity transfers to every minimal interpretation. The converse is the (second-order) surjectivity problem discussed in the paper.

theorem *Deep-to-MinS*:

$(\models^d \varphi) \implies (\forall II \ gg \ GG. \text{MinS.ValM } (\text{MinS.DpToShM } II \ gg \ GG \ \varphi))$
by (*metis* (*mono-tags*, *lifting*) *MinS.FaithfulMD* *ValD-def*)

Consistency check.

lemma *True nitpick[satisfy] oops*

end

6.2 Faithfulness theorems

theory *MSOinHOL-faithfulness*

imports *MSOinHOL-shallow-minimal*
begin

Re-issuing the locale faithfulness theorems at the constants level.

Deep $\longleftrightarrow$ maximal shallow.

theorem $(\langle I, D, E \rangle, g, G \models^s \llbracket \varphi \rrbracket) \longleftrightarrow (\langle I, D, E \rangle, g, G \models^d \varphi)$
using *FaithfulSDlem* .

theorem $(\models^s \llbracket \varphi \rrbracket) \longleftrightarrow (\models^d \varphi)$
using *FaithfulSD* .

Deep $\longleftrightarrow$ minimal shallow, relative to the ranges of the chosen assignments gg and GG .

theorem $(\llbracket \varphi \rrbracket) \longleftrightarrow (\langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^d \varphi)$
using *FaithfulMDlem* .

theorem $(\models^m \llbracket \varphi \rrbracket) \longleftrightarrow (\langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^d \varphi)$
using *FaithfulMD* .

Minimal shallow $\longleftrightarrow$ maximal shallow, again relative to the ranges of gg and GG ; obtained by composing the two preceding bridges.

theorem $(\llbracket \varphi \rrbracket) \longleftrightarrow (\langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^s \llbracket \varphi \rrbracket)$
using *FaithfulMSlem* .

theorem $(\models^m \llbracket \varphi \rrbracket) \longleftrightarrow (\langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^s \llbracket \varphi \rrbracket)$
using *FaithfulMS* .

Global form across all interpretations and the one-directional bridge to full deep validity.

theorem
 $(\forall II \, gg \, GG. (\models^m (\text{MinS.DpToShM } II \, gg \, GG \, \varphi))) = (\forall I \, g \, G. \langle I, \text{Range } g, \text{Range } G \rangle, g, G \models^d \varphi)$
using *FaithfulMS-all* **by** (*simp add: MinS.ValM-def*)

theorem $(\models^d \varphi) \implies (\forall II \, gg \, GG. (\models^m (\text{MinS.DpToShM } II \, gg \, GG \, \varphi)))$
using *Deep-to-MinS* **by** (*simp add: MinS.ValM-def*)

Consistency check.

lemma *True* **nitpick**[*satisfy*] **oops**

end

7 Comprehension

theory *MSOinHOL-comprehension*
imports *MSOinHOL-deep-subst-lemma*
begin

Monadic comprehension as a derived schema: under the full second-order domain ($ValD'$, $E = Univ$) comprehension holds for every φ with X not free; the witness is the set defined by φ .

theorem *comprehension-schema*:
assumes X not-free2-in φ
shows $\models^{d'} (\exists^d_2 X. \forall^d x. ((X^d(x)) \longleftrightarrow^d \varphi))$
unfolding $ValD'$ -def
proof (*intro allI*)
fix I g G
let $?S = \lambda d. \langle I, Univ, Univ \rangle, g[x \leftarrow d], G \models^d \varphi$
have $\langle I, Univ, Univ \rangle, g, G \langle X \leftarrow ?S \rangle \models^d (\forall^d x. ((X^d(x)) \longleftrightarrow^d \varphi))$
using *assms* **by** (*simp add: DefD N12*)
thus $\langle I, Univ, Univ \rangle, g, G \models^d (\exists^d_2 X. \forall^d x. ((X^d(x)) \longleftrightarrow^d \varphi))$
by *auto*
qed

Headline instance (cf. *comprehension-d*): the set of r -self-related individuals exists.

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*

Standard vs. weak (Henkin): *comprehension-schema* needs $E = Univ$ and fails for general $ValD$ —the standard-vs-Henkin gap behind the Loewenheim–Skolem problem, treated in the next section (*Downward Löwenheim–Skolem*).

end

8 Downward Löwenheim–Skolem

8.1 Tarski–Vaught stage lemmas

theory *MSOinHOL-lowenheim-skolem-lemmas*
imports
 $MSOinHOL$ -faithfulness-locale
 $MSOinHOL$ -comprehension
 HOL –Library.Countable-Set
begin

Supporting lemmas for the downward Loewenheim–Skolem construction in *MSOinHOL-lowenheim-skolem*. This file holds the technical machinery (Skolem-witness operators wF / wS , the omega-stage iteration *stage*, monotonicity / countability / common-stage lemmas). The main theorems live in the parent file.

Lemma *coincidence* (generalises *L12* / *N12*): truth depends only on the free variables of φ . Compact structured induction; the binder cases use

IH after extending the agreement through the bound-variable update via *EnvMod-def* / *SetMod-def*.

lemma *coincidence*:

```

assumes  $\bigwedge x. x \text{ free-in } \varphi \implies g \ x = g' \ x$ 
and  $\bigwedge X. X \text{ free2-in } \varphi \implies G \ X = G' \ X$ 
shows  $(\langle I, D, E \rangle, g, G \models^d \varphi) = (\langle I, D, E \rangle, g', G' \models^d \varphi)$ 
using assms
proof (induct  $\varphi$  arbitrary:  $g \ g' \ G \ G'$ )
  case (AtmD  $r \ u \ v$ ) thus ?case by simp
next
  case (PrdD  $X \ z$ ) thus ?case by simp
next
  case (NegD  $\varphi$ )
  have  $(\langle I, D, E \rangle, g, G \models^d \varphi) = (\langle I, D, E \rangle, g', G' \models^d \varphi)$ 
    using NegD.hyps NegD.premis(1,2)
      is-free.simps(3) is-free2.simps(3)
    by presburger
  thus ?case by simp
next
  case (AndD  $\varphi \ \psi$ )
  from AndD.hyps[of g g' G G'] AndD.premis show ?case by simp
next
  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  $d$ 
    using EnvMod-def ExD.hyps ExD.premis(1,2)
      is-free.simps(5) is-free2.simps(5)
    by presburger
  thus ?case by simp
next
  case (ExD2  $Y \ \varphi$ )
  have  $(\langle I, D, E \rangle, g, G \langle Y \leftarrow S \rangle \models^d \varphi) = (\langle I, D, E \rangle, g', G' \langle Y \leftarrow S \rangle \models^d \varphi)$  for  $S$ 
    by (rule ExD2.hyps) (use ExD2.premis in  $\langle \text{auto simp: SetMod-def} \rangle$ )
  thus ?case by simp
qed

```

The formula type $\mathcal{F}$ is countable (datatype over countable atoms).

instance $\mathcal{F} :: \text{countable}$ **by** *countable-datatype*

Pad a finite parameter list to a total assignment (default outside its range).

definition *padA* :: $D \Rightarrow D \text{ list} \Rightarrow (V \Rightarrow D)$
where *padA* $d0 \ xs = (\lambda i. \text{if } i < \text{length } xs \text{ then } xs ! i \text{ else } d0)$

definition *padB* :: $(D \Rightarrow \text{bool}) \Rightarrow (D \Rightarrow \text{bool}) \text{ list} \Rightarrow (V2 \Rightarrow (D \Rightarrow \text{bool}))$
where *padB* $S0 \ Ss = (\lambda i. \text{if } i < \text{length } Ss \text{ then } Ss ! i \text{ else } S0)$

First-order (FO) and second-order (SO) Skolem witness functions wF and wS for a single existential demand of the form $\exists y. \psi$ or $\exists_2 Y. \psi$. A

partial assignment is supplied as a list of free-variable values padded out by defaults d_0 and S_0 (via $padA$ / $padB$). The function then asks whether the existential demand is satisfiable in the big model (I, D, E) ; if so it picks a satisfying witness via the *SOME* operator, and otherwise falls back to the default. These witnesses are the per-step building blocks of the Tarski–Vaught stage construction below, which feeds every existential demand of the formula with a witness whose value comes from the big model but whose name lives in the (countable) hull.

definition $wF ::$

$$\begin{aligned} \mathcal{I} &\Rightarrow (D \Rightarrow bool) \Rightarrow ((D \Rightarrow bool) \Rightarrow bool) \\ &\Rightarrow D \Rightarrow (D \Rightarrow bool) \Rightarrow \mathcal{F} \Rightarrow V \Rightarrow D \text{ list} \\ &\Rightarrow (D \Rightarrow bool) \text{ list} \Rightarrow D \end{aligned}$$

where

$$\begin{aligned} wF \ I \ D \ E \ d0 \ S0 \ \psi \ y \ xs \ Ss = \\ \text{if } (\exists d. D \ d \wedge \langle I, D, E \rangle, (padA \ d0 \ xs)[y \leftarrow d], (padB \ S0 \ Ss) \models^d \psi) \\ \text{then } (SOME \ d. D \ d \wedge \\ \quad \langle I, D, E \rangle, (padA \ d0 \ xs)[y \leftarrow d], (padB \ S0 \ Ss) \models^d \psi) \\ \text{else } d0 \end{aligned}$$

definition $wS ::$

$$\begin{aligned} \mathcal{I} &\Rightarrow (D \Rightarrow bool) \Rightarrow ((D \Rightarrow bool) \Rightarrow bool) \\ &\Rightarrow D \Rightarrow (D \Rightarrow bool) \Rightarrow \mathcal{F} \Rightarrow V2 \Rightarrow D \text{ list} \\ &\Rightarrow (D \Rightarrow bool) \text{ list} \Rightarrow (D \Rightarrow bool) \end{aligned}$$

where

$$\begin{aligned} wS \ I \ D \ E \ d0 \ S0 \ \psi \ Y \ xs \ Ss = \\ \text{if } (\exists S. E \ S \wedge \\ \quad \langle I, D, E \rangle, (padA \ d0 \ xs), (padB \ S0 \ Ss) \langle Y \leftarrow S \rangle \models^d \psi) \\ \text{then } (SOME \ S. E \ S \wedge \\ \quad \langle I, D, E \rangle, (padA \ d0 \ xs), (padB \ S0 \ Ss) \langle Y \leftarrow S \rangle \models^d \psi) \\ \text{else } S0 \end{aligned}$$

lemma $wF\text{-mem}: D \ d0 \implies D \ (wF \ I \ D \ E \ d0 \ S0 \ \psi \ y \ xs \ Ss)$

unfolding $wF\text{-def}$

by (*cases* $\exists d. D \ d \wedge$
 $\langle I, D, E \rangle, (padA \ d0 \ xs)[y \leftarrow d], (padB \ S0 \ Ss) \models^d \psi;$
simp, metis (mono-tags, lifting) someI-ex)

lemma $wS\text{-mem}: E \ S0 \implies E \ (wS \ I \ D \ E \ d0 \ S0 \ \psi \ Y \ xs \ Ss)$

unfolding $wS\text{-def}$

by (*cases* $\exists S. E \ S \wedge$
 $\langle I, D, E \rangle, (padA \ d0 \ xs), (padB \ S0 \ Ss) \langle Y \leftarrow S \rangle \models^d \psi;$
simp, metis (mono-tags, lifting) someI-ex)

Witness-membership specialised to seeds $g \ 0$ / $G \ 0$: no extra $D \ (g \ 0)$ premise.

lemma $wF\text{-mem-seed}:$

$g \ \text{into} \ D \implies D \ (wF \ I \ D \ E \ (g \ 0) \ S0 \ \psi \ y \ xs \ Ss)$

by (*blast intro: wF-mem*)

lemma *wS-mem-seed*:

$G \text{ into } E \implies E (wS \ I \ D \ E \ d_0 \ (G \ 0) \ \psi \ Y \ xs \ Ss)$
by (*blast intro: wS-mem*)

The omega-stage (FO carrier, SO carrier): step $n+1$ adds every wF / wS witness over parameter lists drawn from the carriers at stage n .

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

$\text{stage } I \ D \ E \ g \ G \ 0 = (\text{Range } g, \text{Range } G)$
 $\mid \text{stage } I \ D \ E \ g \ G \ (\text{Suc } n) =$
 $((\lambda d. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ d$
 $\vee (\exists \psi \ y \ xs \ Ss.$
 $(\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ x)$
 $\wedge (\forall S \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S)$
 $\wedge d = wF \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ y \ xs \ Ss)),$
 $(\lambda S. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S$
 $\vee (\exists \psi \ Y \ xs \ Ss.$
 $(\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ x)$
 $\wedge (\forall S' \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S')$
 $\wedge S = wS \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ Y \ xs \ Ss)))$

Helper: countability of a binary product (from *countable-SIGMA*).

lemma *countable-Times2*:

$\text{countable } A \implies \text{countable } B \implies \text{countable } (A \times B)$
using *countable-SIGMA*[of $A \ \lambda \cdot. B$] **by** (*simp add: Sigma-def*)

An explicit Skolem witness produced from parameters drawn from stage n sits in stage $\text{Suc } n$ (FO / SO). Direct from the second clause of *stage*.

lemma *fst-stage-Suc-intro*:

$\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ N) \ x \implies$
 $\forall S \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ N) \ S \implies$
 $\text{fst } (\text{stage } I \ D \ E \ g \ G \ (\text{Suc } N)) (wF \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ y \ xs \ Ss)$
by (*simp, blast*)

lemma *snd-stage-Suc-intro*:

$\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ N) \ x \implies$
 $\forall S \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ N) \ S \implies$
 $\text{snd } (\text{stage } I \ D \ E \ g \ G \ (\text{Suc } N)) (wS \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ Y \ xs \ Ss)$
by (*simp, blast*)

The next two lemmas are instances of the coincidence lemma in which the assignments a , b are first restricted to the free-variable prefix of ψ (the indices $[0..<\text{fresh } \psi]$ and $[0..<\text{fresh2 } \psi]$) and then re-extended to total assignments via padA / padB with defaults d_0 , S_0 . Because ψ can only look at the variables it actually uses, the value of ψ under the padded assignment

coincides with its value under the original one. This is precisely the rewrite needed for the Tarski–Vaught step: the truth of an existential demand on the partial (padded) assignment is the truth of the same demand on the original assignment, so a witness chosen for the padded form automatically refutes / verifies the original form, and the truth transfer through one stage of the hull construction goes through by *simp* alone.

lemma *pad-truth-eq-FO*:

$(\langle I, D, E \rangle, (\text{padA } d_0 (\text{map } a [0..<\text{fresh } \psi]))[y \leftarrow d], (\text{padB } S_0 (\text{map } b [0..<\text{fresh2 } \psi])) \models^d \psi) = (\langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi)$
by (*rule coincidence*)
(auto dest!: L6 N6 simp: EnvMod-def padA-def padB-def)

lemma *pad-truth-eq-SO*:

$(\langle I, D, E \rangle, \text{padA } d_0 (\text{map } a [0..<\text{fresh } \psi]), (\text{padB } S_0 (\text{map } b [0..<\text{fresh2 } \psi])) \langle Y \leftarrow S \rangle \models^d \psi) = (\langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi)$
by (*rule coincidence*)
(auto dest!: L6 N6 simp: SetMod-def padA-def padB-def)

The Skolem witness wF / wS realizes its demand whenever one exists in (D, E) : membership in D / E plus the truth on the padded assignment. Direct from *someI-ex*.

lemma *wF-realizes*:

assumes $\exists d. D \ d \wedge$
 $\langle I, D, E \rangle, (\text{padA } d_0 \ xs)[y \leftarrow d], (\text{padB } S_0 \ Ss) \models^d \psi$
shows $D \ (wF \ I \ D \ E \ d_0 \ S_0 \ \psi \ y \ xs \ Ss)$
 $\wedge (\langle I, D, E \rangle, (\text{padA } d_0 \ xs)[y \leftarrow wF \ I \ D \ E \ d_0 \ S_0 \ \psi \ y \ xs \ Ss],$
 $(\text{padB } S_0 \ Ss) \models^d \psi)$
using *someI-ex*[*OF assms*] *assms* **unfolding** *wF-def* **by** *simp*

lemma *wS-realizes*:

assumes $\exists S. E \ S \wedge$
 $\langle I, D, E \rangle, (\text{padA } d_0 \ xs), (\text{padB } S_0 \ Ss) \langle Y \leftarrow S \rangle \models^d \psi$
shows $E \ (wS \ I \ D \ E \ d_0 \ S_0 \ \psi \ Y \ xs \ Ss)$
 $\wedge (\langle I, D, E \rangle, (\text{padA } d_0 \ xs),$
 $(\text{padB } S_0 \ Ss) \langle Y \leftarrow wS \ I \ D \ E \ d_0 \ S_0 \ \psi \ Y \ xs \ Ss \rangle$
 $\models^d \psi)$
using *someI-ex*[*OF assms*] *assms* **unfolding** *wS-def* **by** *simp*

Full Tarski–Vaught step (FO / SO): from an existential over (D, E) reachable through a, b , the wF / wS witness lives in D / E and satisfies the formula relative to a, b . Core lemma for the *TVfo* / *TVso* of *skolem-hull*.

lemma *skolem-step-FO*:

assumes $\exists d. D \ d \wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi$
shows $D \ (wF \ I \ D \ E \ d_0 \ S_0 \ \psi \ y$
 $(\text{map } a [0..<\text{fresh } \psi]) (\text{map } b [0..<\text{fresh2 } \psi]))$
 $\wedge (\langle I, D, E \rangle, a[y \leftarrow wF \ I \ D \ E \ d_0 \ S_0 \ \psi \ y$
 $(\text{map } a [0..<\text{fresh } \psi])$
 $(\text{map } b [0..<\text{fresh2 } \psi])),$

$$b \models^d \psi$$

proof –

let $?xs = \text{map } a [0..<\text{fresh } \psi]$
and $?Ss = \text{map } b [0..<\text{fresh2 } \psi]$
have $\exists d. D \ d \wedge \langle I, D, E \rangle, (\text{padA } d_0 \ ?xs)[y \leftarrow d], (\text{padB } S_0 \ ?Ss) \models^d \psi$
using *assms* **by** (*simp add: pad-truth-eq-FO*)
from *wF-realizes[OF this]* **show** *?thesis*
by (*simp add: pad-truth-eq-FO*)

qed

lemma *skolem-step-SO*:

assumes $\exists S. E \ S \wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi$
shows $E \ (wS \ I \ D \ E \ d_0 \ S_0 \ \psi \ Y$
 $\quad (\text{map } a [0..<\text{fresh } \psi]) \ (\text{map } b [0..<\text{fresh2 } \psi]))$
 $\wedge \langle \langle I, D, E \rangle, a,$
 $\quad b \langle Y \leftarrow wS \ I \ D \ E \ d_0 \ S_0 \ \psi \ Y$
 $\quad (\text{map } a [0..<\text{fresh } \psi])$
 $\quad (\text{map } b [0..<\text{fresh2 } \psi]) \rangle$
 $\models^d \psi)$

proof –

let $?xs = \text{map } a [0..<\text{fresh } \psi]$
and $?Ss = \text{map } b [0..<\text{fresh2 } \psi]$
have $\exists S. E \ S \wedge \langle I, D, E \rangle, (\text{padA } d_0 \ ?xs), (\text{padB } S_0 \ ?Ss) \langle Y \leftarrow S \rangle \models^d \psi$
using *assms* **by** (*simp add: pad-truth-eq-SO*)
from *wS-realizes[OF this]* **show** *?thesis*
by (*simp add: pad-truth-eq-SO*)

qed

The (*Suc n*)-stage is contained in the *n*-stage plus the image of *wF* / *wS* over the four-fold product $\mathcal{F} \times V \times \text{Lists}_{\mathcal{D}}(n) \times \text{Lists}_{\mathcal{P}}(n)$, where the first two factors range over all formulas and all binder variables, and the last two over lists of FO domain elements resp. SO predicates that already live in the *n*-th stage. Used to derive countability of each stage by induction.

lemma *fst-stage-Suc-subset*:

$\{d. \text{fst } (\text{stage } I \ D \ E \ g \ G \ (\text{Suc } n)) \ d\}$
 $\subseteq \{d. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ d\}$
 $\cup (\lambda(\psi, y, xs, Ss). wF \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ y \ xs \ Ss) \cdot$
 $\quad (UNIV \times UNIV$
 $\quad \times \{xs. \text{set } xs \subseteq \{d. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ d\}\}$
 $\quad \times \{Ss. \text{set } Ss \subseteq \{S. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S\}\})$
 $(\text{is } - \subseteq ?R)$

proof (*rule subsetI, simp only: mem-Collect-eq*)

fix *d* **assume** *fst (stage I D E g G (Suc n)) d*

then consider

fst (stage I D E g G n) d

| *ψ y xs Ss*

where $\forall x \in \text{set } xs. \text{fst } (\text{stage } I \ D \ E \ g \ G \ n) \ x$
 $\forall S \in \text{set } Ss. \text{snd } (\text{stage } I \ D \ E \ g \ G \ n) \ S$
 $d = wF \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ y \ xs \ Ss$

by auto
 thus $d \in ?R$
 by cases (auto intro!: rev-image-eqI[where $x=(-, -, -, -)$])
 qed

lemma *snd-stage-Suc-subset*:

$\{S. \text{snd } (\text{stage } I D E g G (\text{Suc } n)) S\}$
 $\subseteq \{S. \text{snd } (\text{stage } I D E g G n) S\}$
 $\cup (\lambda(\psi, Y, xs, Ss). wS I D E (g 0) (G 0) \psi Y xs Ss) '$
 $(UNIV \times UNIV$
 $\times \{xs. \text{set } xs \subseteq \{d. \text{fst } (\text{stage } I D E g G n) d\}\}$
 $\times \{Ss. \text{set } Ss \subseteq \{S. \text{snd } (\text{stage } I D E g G n) S\}\})$
 (is $\subseteq ?R$)

proof (rule subsetI, simp only: mem-Collect-eq)

fix S **assume** $\text{snd } (\text{stage } I D E g G (\text{Suc } n)) S$

then consider

$\text{snd } (\text{stage } I D E g G n) S$

| $\psi Y xs Ss$

where $\forall x \in \text{set } xs. \text{fst } (\text{stage } I D E g G n) x$

$\forall S' \in \text{set } Ss. \text{snd } (\text{stage } I D E g G n) S'$

$S = wS I D E (g 0) (G 0) \psi Y xs Ss$

by auto

thus $S \in ?R$

by cases (auto intro!: rev-image-eqI[where $x=(-, -, -, -)$])

qed

Monotonicity of the *stage* chain: a single-step extension lifts to arbitrary suffixes.

lemma *fst-stage-mono*:

$m \leq n \implies \text{fst } (\text{stage } I D E g G m) d \implies \text{fst } (\text{stage } I D E g G n) d$

by (induct n) (auto simp del: stage.simps simp: stage.simps(2) le-Suc-eq)

lemma *snd-stage-mono*:

$m \leq n \implies \text{snd } (\text{stage } I D E g G m) S \implies \text{snd } (\text{stage } I D E g G n) S$

by (induct n) (auto simp del: stage.simps simp: stage.simps(2) le-Suc-eq)

Finitely many parameters in the omega-union all share a common stage.

lemma *common-stage-lifted*:

assumes $hx: \forall x \in \text{set } xs. \exists n. \text{fst } (\text{stage } I D E g G n) x$

and $hS: \forall S \in \text{set } Ss. \exists n. \text{snd } (\text{stage } I D E g G n) S$

shows $\exists N. (\forall x \in \text{set } xs. \text{fst } (\text{stage } I D E g G N) x)$

$\wedge (\forall S \in \text{set } Ss. \text{snd } (\text{stage } I D E g G N) S)$

proof –

obtain fX **where** $fX: \forall x \in \text{set } xs. \text{fst } (\text{stage } I D E g G (fX x)) x$

using hx **by** metis

obtain fS **where** $fS: \forall S \in \text{set } Ss. \text{snd } (\text{stage } I D E g G (fS S)) S$

using hS **by** metis

let $?N = \text{Max } (\text{insert } 0 (fX ' \text{set } xs \cup fS ' \text{set } Ss))$

have $bX: fX x \leq ?N$ **if** $x \in \text{set } xs$ **for** x

using *that* **by** (*simp add: Max-ge*)
have $bS: fS \ S \leq ?N$ **if** $S \in \text{set } Ss$ **for** S
using *that* **by** (*simp add: Max-ge*)
from $bX \ fX$ *fst-stage-mono*
have $\forall x \in \text{set } xs. \text{fst}(\text{stage } I \ D \ E \ g \ G \ ?N) \ x$ **by** *blast*
moreover from $bS \ fS$ *snd-stage-mono*
have $\forall S \in \text{set } Ss. \text{snd}(\text{stage } I \ D \ E \ g \ G \ ?N) \ S$ **by** *blast*
ultimately show *?thesis* **by** *blast*
qed

A wF / wS witness built from hull-valued parameters is itself in the hull.
Combines *common-stage-lifted* with the single-Suc-step intros.

lemma *wF-in-omega-union*:
assumes $\forall x \in \text{set } xs. \exists n. \text{fst}(\text{stage } I \ D \ E \ g \ G \ n) \ x$
and $\forall S \in \text{set } Ss. \exists n. \text{snd}(\text{stage } I \ D \ E \ g \ G \ n) \ S$
shows $\exists n. \text{fst}(\text{stage } I \ D \ E \ g \ G \ n)$
 $(wF \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ y \ xs \ Ss)$
using *common-stage-lifted[OF assms]*
by (*meson fst-stage-Suc-intro*)

lemma *wS-in-omega-union*:
assumes $\forall x \in \text{set } xs. \exists n. \text{fst}(\text{stage } I \ D \ E \ g \ G \ n) \ x$
and $\forall S \in \text{set } Ss. \exists n. \text{snd}(\text{stage } I \ D \ E \ g \ G \ n) \ S$
shows $\exists n. \text{snd}(\text{stage } I \ D \ E \ g \ G \ n)$
 $(wS \ I \ D \ E \ (g \ 0) \ (G \ 0) \ \psi \ Y \ xs \ Ss)$
using *common-stage-lifted[OF assms]*
by (*meson snd-stage-Suc-intro*)

Tarski-Vaught closure of the omega-union: combines *skolem-step-FO / skolem-step-SO* with *wF-in-omega-union / wS-in-omega-union*.

lemma *stage-TV-FO*:
assumes $a \text{ into } (\lambda d. \exists n. \text{fst}(\text{stage } I \ D \ E \ g \ G \ n) \ d)$
and $b \text{ into } (\lambda S. \exists n. \text{snd}(\text{stage } I \ D \ E \ g \ G \ n) \ S)$
and $\exists d. D \ d \wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi$
shows $\exists d. (\exists n. \text{fst}(\text{stage } I \ D \ E \ g \ G \ n) \ d)$
 $\wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi$
using *skolem-step-FO[OF assms(3), of g 0 G 0]*
 $wF\text{-in-}\omega\text{-union}[\text{of map } a \ [0..<\text{fresh } \psi] \ I \ D \ E \ g \ G]$
 $\text{map } b \ [0..<\text{fresh2 } \psi] \ \psi \ y]$
 $\text{assms}(1, 2)$
by *auto*

lemma *stage-TV-SO*:
assumes $a \text{ into } (\lambda d. \exists n. \text{fst}(\text{stage } I \ D \ E \ g \ G \ n) \ d)$
and $b \text{ into } (\lambda S. \exists n. \text{snd}(\text{stage } I \ D \ E \ g \ G \ n) \ S)$
and $\exists S. E \ S \wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi$
shows $\exists S. (\exists n. \text{snd}(\text{stage } I \ D \ E \ g \ G \ n) \ S)$
 $\wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi$
using *skolem-step-SO[OF assms(3), of g 0 G 0]*

```

      wS-in-omega-union[of map a [0..fresh  $\psi$ ]  $I D E g G$ 
                        map b [0..fresh2  $\psi$ ]  $\psi Y$ ]
    assms(1,2)
  by auto

```

Inductive step for stage countability: each *Suc*-stage is countable provided both *n*-stages are.

lemma *cntDE-step*:

```

  assumes cD: countable {d. fst (stage  $I D E g G n$ ) d}
    and cE: countable {S. snd (stage  $I D E g G n$ ) S}
  shows countable {d. fst (stage  $I D E g G (Suc\ n)$ ) d}
    and countable {S. snd (stage  $I D E g G (Suc\ n)$ ) S}
  proof -
    let ?A = {xs. set xs  $\subseteq$  {d. fst (stage  $I D E g G n$ ) d}}
    let ?B = {Ss. set Ss  $\subseteq$  {S. snd (stage  $I D E g G n$ ) S}}
    have lDE: countable ?A countable ?B
      apply (metis cD countable-lists lists-eq-set)
      by (metis cE countable-lists lists-eq-set)
    hence cT:
      countable ((UNIV::F set)  $\times$  (UNIV::V set)  $\times$  ?A  $\times$  ?B)
      countable ((UNIV::F set)  $\times$  (UNIV::V2 set)  $\times$  ?A  $\times$  ?B)
      by (auto intro!: countable-Times2)
    hence prods:
      countable (( $\lambda(\psi,y,xs,Ss).$  wF  $I D E (g\ 0) (G\ 0) \psi y xs Ss$ ) ‘
        ((UNIV::F set)  $\times$  (UNIV::V set)  $\times$  ?A  $\times$  ?B))
      countable (( $\lambda(\psi,Y,xs,Ss).$  wS  $I D E (g\ 0) (G\ 0) \psi Y xs Ss$ ) ‘
        ((UNIV::F set)  $\times$  (UNIV::V2 set)  $\times$  ?A  $\times$  ?B))
      by (auto intro: countable-image)
    from prods(1) cD fst-stage-Suc-subset[of  $I D E g G n$ ]
      show countable {d. fst (stage  $I D E g G (Suc\ n)$ ) d}
      by (meson countable-Un countable-subset)
    from prods(2) cE snd-stage-Suc-subset[of  $I D E g G n$ ]
      show countable {S. snd (stage  $I D E g G (Suc\ n)$ ) S}
      by (meson countable-Un countable-subset)
  qed

```

Stage carriers stay inside *D* / *E* (by induction on the stage; *wF*-mem-*seed* / *wS*-mem-*seed* handle the witness case).

lemma *fst-stage-subD*:

```

  g into D  $\implies$  fst (stage  $I D E g G n$ ) d  $\implies D\ d$ 
  by (induct n arbitrary: d) (auto simp: wF-mem-seed)

```

lemma *snd-stage-subE*:

```

  G into E  $\implies$  snd (stage  $I D E g G n$ ) S  $\implies E\ S$ 
  by (induct n arbitrary: S) (auto simp: wS-mem-seed)

```

Every stage is countable; hence so is the omega-union of stages. Pure consequence of *cntDE-step*.

lemma *stage-countable*:

```

countable {d. fst (stage I D E g G n) d}
  ∧ countable {S. snd (stage I D E g G n) S}
proof (induct n)
  case 0
  have {d. fst (stage I D E g G 0) d} = range g
    {S. snd (stage I D E g G 0) S} = range G
    by auto
  thus ?case by simp
next
  case (Suc n) thus ?case
    using cntDE-step[of I D E g G n] by simp
qed

```

```

lemma stage-omega-countable:
countable {d. ∃ n. fst (stage I D E g G n) d}
countable {S. ∃ n. snd (stage I D E g G n) S}
proof -
  have eqs:
    {d. ∃ n. fst (stage I D E g G n) d}
    = (⋃ n. {d. fst (stage I D E g G n) d})
    {S. ∃ n. snd (stage I D E g G n) S}
    = (⋃ n. {S. snd (stage I D E g G n) S})
    by auto
  show countable {d. ∃ n. fst (stage I D E g G n) d}
    countable {S. ∃ n. snd (stage I D E g G n) S}
    unfolding eqs using stage-countable by (auto intro: countable-UN)
qed

```

Reindex one sort onto a countable nonempty sub-domain, keeping the values on a finite “guarded” prefix specified by a predicate *free* bounded by *fr*.

```

lemma reindex-one:
fixes free :: nat ⇒ bool and fr :: nat
and D0 :: 'a ⇒ bool and g :: nat ⇒ 'a
assumes countable {x. D0 x} and D0 d0
and Range g ⊆ D0
and bound: ∧ x. free x ⇒ x < fr
obtains g' where Range g' = D0
and ∧ x. free x ⇒ g' x = g x
proof -
let ?D = {x. D0 x}
have hD: ?D ≠ {}
and re: range (from-nat-into ?D) = ?D
and mem: D0 (from-nat-into ?D k) for k
using assms(2) range-from-nat-into[OF - assms(1)]
  from-nat-into
by auto
define g' where
  g' n = (if free n then g n else from-nat-into ?D (n - fr))

```

```

for  $n$ 
have  $\text{Range } g' = D_0$ 
proof (rule ext, rule iffI)
  fix  $d$  assume  $\text{Range } g' d$ 
  thus  $D_0 d$ 
    using  $\text{assms}(3) \text{ mem}$ 
    by (auto simp: g'-def split: if-split-asm)
next
  fix  $d$  assume  $D_0 d$ 
  hence  $d \in \text{range } (\text{from-nat-into } ?D)$  using  $re$  by simp
  then obtain  $k$  where  $\text{from-nat-into } ?D k = d$  by auto
  moreover have  $\neg \text{free } (fr + k)$  using bound by force
  ultimately have  $g' (fr + k) = d$  by (simp add: g'-def)
  thus  $\text{Range } g' d$  by auto
qed
moreover have  $\bigwedge x. \text{free } x \implies g' x = g x$ 
  by (simp add: g'-def)
ultimately show ?thesis using that by blast
qed

```

Reindex (both sorts): reindex the (countably many) variables to surject onto the countable D_0 , E_0 , while agreeing with the originals on the finitely many free variables of φ . Two applications of *reindex-one*.

lemma *reindex*:

```

assumes countable  $\{d. D_0 d\}$  and  $D_0 d_0$ 
and countable  $\{S. E_0 S\}$  and  $E_0 S_0$ 
and  $\text{Range } g \subseteq D_0$  and  $\text{Range } G \subseteq E_0$ 
obtains  $g' G'$ 
  where  $\text{Range } g' = D_0$  and  $\text{Range } G' = E_0$ 
  and  $\bigwedge x. x \text{ free-in } \varphi \implies g' x = g x$ 
  and  $\bigwedge X. X \text{ free2-in } \varphi \implies G' X = G X$ 
proof –
  obtain  $g'$  where  $g'$ :
     $\text{Range } g' = D_0$ 
     $\bigwedge x. x \text{ free-in } \varphi \implies g' x = g x$ 
    using reindex-one[OF assms(1,2,5),
      of  $\lambda x. x \text{ free-in } \varphi \text{ fresh } \varphi$ ]
      L6
    by blast
  obtain  $G'$  where  $G'$ :
     $\text{Range } G' = E_0$ 
     $\bigwedge X. X \text{ free2-in } \varphi \implies G' X = G X$ 
    using reindex-one[OF assms(3,4,6),
      of  $\lambda X. X \text{ free2-in } \varphi \text{ fresh2 } \varphi$ ]
      N6
    by blast
  show ?thesis using that  $g' G'$  by blast
qed

```

Reindex + coincidence in one step: obtain g'/G' surjecting onto the

countable D_0/E_0 whose induced truth on φ coincides with the original.

lemma *reindex-coincide*:

assumes *countable* $\{d. D_0 d\}$ **and** *countable* $\{S. E_0 S\}$

and *Range* $g \subseteq D_0$ **and** *Range* $G \subseteq E_0$

obtains $g' G'$

where *Range* $g' = D_0$ **and** *Range* $G' = E_0$

and $(\langle I, D_0, E_0 \rangle, g, G \models^d \varphi)$
 $= (\langle I, D_0, E_0 \rangle, g', G' \models^d \varphi)$

proof –

have $nD: D_0 (g \ 0)$ **and** $nE: E_0 (G \ 0)$

using *assms*(3,4) **by** *auto*

obtain $g' G'$ **where** $rg: \text{Range } g' = D_0$ **and** $rG: \text{Range } G' = E_0$

and $ag: \bigwedge x. x \text{ free-in } \varphi \implies g' x = g x$

and $aG: \bigwedge X. X \text{ free2-in } \varphi \implies G' X = G X$

using *reindex*[*OF* *assms*(1) *nD* *assms*(2) *nE* *assms*(3,4)] **by** *blast*

have $(\langle I, D_0, E_0 \rangle, g, G \models^d \varphi) = (\langle I, D_0, E_0 \rangle, g', G' \models^d \varphi)$

by (*rule coincidence*) (*simp-all add: ag aG*)

thus *thesis* **using** that $rg \ rG$ **by** *blast*

qed

end

8.2 The theorem and its corollaries

theory *MSOinHOL-lowenheim-skolem*

imports *MSOinHOL-lowenheim-skolem-lemmas*

begin

Two-sorted elementary substructure (Tarski–Vaught form).

definition *ElementarySubstructure*

$(\langle -, -, \rangle \subseteq_E \langle -, -, \rangle)$

where

$\langle I', D', E' \rangle \subseteq_E \langle I, D, E \rangle \equiv$

$D' \subseteq D \wedge E' \subseteq E$

$\wedge (\forall g \ G \ \varphi. \ G \text{ into } E' \longrightarrow g \text{ into } D'$

$\longrightarrow ((\langle I, D, E \rangle, g, G \models^d \varphi) = (\langle I', D', E' \rangle, g, G \models^d \varphi)))$

Minimal interpretation whose range model is elementary in a larger model.

locale *MinS-ES* = *MinS* +

fixes $DD :: \mathcal{D}$ **and** $EE :: \mathcal{P}$

assumes $ES: \langle II, \text{Range } gg, \text{Range } GG \rangle \subseteq_E \langle II, DD, EE \rangle$

begin

lemma *N-valid-ES*:

$\langle II, DD, EE \rangle, gg, GG \models^d \varphi = \langle II, \text{Range } gg, \text{Range } GG \rangle, gg, GG \models^d \varphi$

using *ES ElementarySubstructure-def* **by** (*smt* (*verit*))

end

Specialisation to the standard model $\langle II, Univ, Univ \rangle$.

locale *MinS-ES-Univ* = *MinS-ES* *II* *gg* *GG* *Univ* *Univ* **for** *II* *gg* *GG*
begin

lemma *N-valid-ES-Univ*:
 $\langle II, Univ, Univ \rangle, gg, GG \models^d \varphi = \langle II, Range\ gg, Range\ GG \rangle, gg, GG \models^d \varphi$
using *N-valid-ES* .

end

Range-relative validity: the right-hand side of *FaithfulMS-all*.

definition *RangeValid* ($\models^r$ - 9)
where $\models^r \varphi \equiv \forall I\ g\ G. \langle I, Range\ g, Range\ G \rangle, g, G \models^d \varphi$

lemma *into-Range*: *f into Range f*
by *auto*

Easy direction.

lemma *ValD-imp-RangeValid*: $\models^d \varphi \implies \models^r \varphi$
unfolding *RangeValid-def* *ValD-def* **using** *into-Range* **by** *smt*

Truth preservation across a Tarski–Vaught-closed sub-pair $(D0, E0)$.

lemma *truth-pres*:
assumes *subD*: $D_0 \subseteq D$ **and** *subE*: $E_0 \subseteq E$
and *TVfo*:
 $\bigwedge y\ \psi\ a\ b.$
 $a\ into\ D_0 \implies b\ into\ E_0 \implies$
 $(\exists d. D\ d \wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi) \implies$
 $(\exists d. D_0\ d \wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi)$
and *TVso*:
 $\bigwedge Y\ \psi\ a\ b.$
 $a\ into\ D_0 \implies b\ into\ E_0 \implies$
 $(\exists S. E\ S \wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi) \implies$
 $(\exists S. E_0\ S \wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi)$
and *aD*: *a into D₀* **and** *bE*: *b into E₀*
shows $(\langle I, D_0, E_0 \rangle, a, b \models^d \psi) = (\langle I, D, E \rangle, a, b \models^d \psi)$
using *aD* *bE*

proof (*induct* ψ *arbitrary*: *a b*)
case (*ExD* *y* ψ)
thus *?case*
using *ExD.hyps*[*of* *a*[*y* $\leftarrow$ -] *b*] *TVfo*[*of* *a* *b* *y* ψ] *subD*
by (*force simp: EnvMod-def*)

next
case (*ExD2* *Y* ψ)
thus *?case*
using *ExD2.hyps*[*of* *a* *b* $\langle Y \leftarrow - \rangle$] *TVso*[*of* *a* *b* *Y* ψ] *subE*
by (*force simp: SetMod-def*)

qed *simp-all*

Countable seed-form Tarski–Vaught hull: extends any countable nonempty $(D_0, E_0) \subseteq (D, E)$ to a countable TV-closed pair (N, M) between them.

lemma *skolem-hull*:

assumes $D_0 \subseteq D$ **and** $E_0 \subseteq E$
and $\exists x. D_0 x$ **and** $\exists S. E_0 S$
and countable $\{d. D_0 d\}$ **and** countable $\{d. E_0 d\}$
obtains $N M$
where $N \subseteq D$ **and** $M \subseteq E$
and $D_0 \subseteq N$ **and** $E_0 \subseteq M$
and countable $\{d. N d\}$ **and** countable $\{S. M S\}$
and $\bigwedge y \psi a b.$
 $a \text{ into } N \implies b \text{ into } M \implies$
 $(\exists d. D d \wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi) \implies$
 $(\exists d. N d \wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi)$
and $\bigwedge Y \psi a b.$
 $a \text{ into } N \implies b \text{ into } M \implies$
 $(\exists S. E S \wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi) \implies$
 $(\exists S. M S \wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi)$

proof –

obtain $g :: \mathcal{E}$ **where** $g: \text{Range } g = D_0$
by (*metis* (*mono-tags*, *lifting*) *Collect-inj*
assms(3,5) *empty-Collect-eq full-SetCompr-eq*
range-from-nat-into)
obtain $G :: \mathcal{G}$ **where** $G: \text{Range } G = E_0$
by (*metis* (*mono-tags*, *lifting*) *Collect-inj*
assms(4,6) *empty-Collect-eq full-SetCompr-eq*
range-from-nat-into)
define Dh **where** $Dh: Dh = (\lambda d. \exists n. \text{fst } (\text{stage } I D E g G n) d)$
define Eh **where** $Eh: Eh = (\lambda S. \exists n. \text{snd } (\text{stage } I D E g G n) S)$
have *base*:
 $\text{Range } g d \implies \text{fst } (\text{stage } I D E g G 0) d$
 $\text{Range } G S \implies \text{snd } (\text{stage } I D E g G 0) S$
for $d S$
by *simp-all*
have $RgDh: \text{Range } g \subseteq Dh$
and $RGEh: \text{Range } G \subseteq Eh$
using *base unfolding Dh Eh by blast+*
have $DhsubD: Dh \subseteq D$
using *fst-stage-subD assms*(1)
unfolding Dh **by** (*metis* (*full-types*) g)
have $EhsubE: Eh \subseteq E$
using *snd-stage-subE assms*(2)
unfolding Eh **by** (*metis* (*mono-tags*, *lifting*) G)
have $cDh: \text{countable } \{d. Dh d\}$
and $cEh: \text{countable } \{S. Eh S\}$
unfolding $Dh Eh$ **by** (*rule stage-omega-countable*)
have *TVfo*:
 $a \text{ into } Dh \implies b \text{ into } Eh \implies$

$(\exists d. D \ d \wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi) \implies$
 $(\exists d. Dh \ d \wedge \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi)$
for $y \ \psi \ a \ b$
unfolding $Dh \ Eh$ **by** (rule stage-TV-FO)
have $TVso$:
 $a \text{ into } Dh \implies b \text{ into } Eh \implies$
 $(\exists S. E \ S \wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi) \implies$
 $(\exists S. Eh \ S \wedge \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi)$
for $Y \ \psi \ a \ b$
unfolding $Dh \ Eh$ **by** (rule stage-TV-SO)
show ?thesis
using $DhsubD \ EhsubE \ G \ RGEh \ RgDh \ TVfo \ TVso \ cDh \ cEh \ g$ **that**
by force
qed

Downward Löwenheim–Skolem: TV-closure is upgraded to genuine elementarity via *truth-pres.*

theorem *DownwardLowenheimSkolem*:
assumes $D_0 \subseteq D$ **and** $E_0 \subseteq E$
and $\exists x. D_0 \ x$ **and** $\exists x. E_0 \ x$
and countable $\{d. D_0 \ d\}$ **and** countable $\{S. E_0 \ S\}$
shows $\exists N \ M$.
 $D_0 \subseteq N \wedge$ countable $\{d. N \ d\}$
 $\wedge E_0 \subseteq M \wedge$ countable $\{S. M \ S\}$
 $\wedge \langle I, N, M \rangle \subseteq_E \langle I, D, E \rangle$
proof –
obtain $N \ M$
where $N \subseteq D \ M \subseteq E$
 $D_0 \subseteq N \ E_0 \subseteq M$
countable $\{d. N \ d\}$ countable $\{S. M \ S\}$
 $\bigwedge y \ \psi \ a \ b$.
 $a \text{ into } N \implies b \text{ into } M \implies$
 $\exists d:D. \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi \implies$
 $\exists d:N. \langle I, D, E \rangle, a[y \leftarrow d], b \models^d \psi$
 $\bigwedge Y \ \psi \ a \ b$.
 $a \text{ into } N \implies b \text{ into } M \implies$
 $\exists S:E. \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi \implies$
 $\exists S:M. \langle I, D, E \rangle, a, b \langle Y \leftarrow S \rangle \models^d \psi$
using skolem-hull[of $D_0 \ D \ E_0 \ E$, OF assms] **by** blast
hence $D_0 \subseteq N \wedge$ countable $\{d. N \ d\}$
 $\wedge E_0 \subseteq M \wedge$ countable $\{S. M \ S\}$
 $\wedge \langle I, N, M \rangle \subseteq_E \langle I, D, E \rangle$
using ElementarySubstructure-def truth-pres
skolem-hull[of $D_0 \ D \ E_0 \ E$, OF assms]
by auto
thus ?thesis **by** blast
qed

Strongest faithfulness: standard validity = minimal validity over elementary interpretations.

theorem *Deep'-to-MinS*:

$$(\models^{d'} \varphi) = (\forall II \text{ } gg \text{ } GG. \text{ } MinS\text{-}ES\text{-}Univ \text{ } II \text{ } gg \text{ } GG \longrightarrow MinS.ValM (MinS.DpToShM \text{ } II \text{ } gg \text{ } GG \text{ } \varphi))$$

proof

assume $\models^{d'} \varphi$

thus $\forall II \text{ } gg. \text{ } MinS\text{-}ES\text{-}Univ \text{ } II \text{ } gg \subseteq (\lambda GG. \text{ } MinS.ValM (MinS.DpToShM \text{ } II \text{ } gg \text{ } GG \text{ } \varphi))$

using *MinS.FaithfulMD MinS-ES.N-valid-ES MinS-ES-Univ-def ValD'-def*

by *blast*

next

assume $A: \forall II \text{ } gg \text{ } GG.$

$$MinS\text{-}ES\text{-}Univ \text{ } II \text{ } gg \text{ } GG \longrightarrow MinS.ValM (MinS.DpToShM \text{ } II \text{ } gg \text{ } GG \text{ } \varphi)$$

show $\models^{d'} \varphi$

unfolding *ValD'-def*

proof (*safe*)

fix $I :: \mathcal{I}$ **and** $g :: \mathcal{E}$ **and** $G :: \mathcal{G}$

have $Range \text{ } g \subseteq Univ \text{ } Range \text{ } G \subseteq Univ$

$\exists x. \text{ } Range \text{ } g \text{ } x \exists x. \text{ } Range \text{ } G \text{ } x$

$countable \{d. \text{ } MSOinHOL\text{-}preliminaries.Range \text{ } g \text{ } d\}$

$countable \{S. \text{ } MSOinHOL\text{-}preliminaries.Range \text{ } G \text{ } S\}$

by (*auto simp add: full-SetCompr-eq*)

then obtain $N \text{ } M$

where $\gamma: Range \text{ } g \subseteq N \text{ } countable \{d. \text{ } N \text{ } d\}$

$Range \text{ } G \subseteq M \text{ } countable \{S. \text{ } M \text{ } S\}$

and $ES: \langle I, N, M \rangle \subseteq_E \langle I, Univ, Univ \rangle$

using *DownwardLowenheimSkolem*

[where $D_0 = Range \text{ } g$ **and** $E_0 = Range \text{ } G$

and $D = Univ$ **and** $E = Univ]$

by *metis*

then obtain $g' \text{ } G'$

where $g': Range \text{ } g' = N$ **and** $G': Range \text{ } G' = M$

and $g'G'$:

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

$= (\langle I, N, M \rangle, g, G \models^d \varphi)$

using *reindex* **by** (*smt (verit, best) reindex-coincide*)

hence $\langle I, Range \text{ } g', Range \text{ } G' \rangle \subseteq_E \langle I, Univ, Univ \rangle$

using *ES* **by** *presburger*

then interpret *MinS-ES-Univ* $I \text{ } g' \text{ } G'$

by (*simp add: MinS-ES.intro MinS-ES-Univ-def*)

have $MinS.ValM (MinS.DpToShM \text{ } I \text{ } g' \text{ } G' \text{ } \varphi)$

using *A MinS-ES-Univ-axioms* **by** *blast*

hence $\langle I, N, M \rangle, g, G \models^d \varphi$

using *FaithfulMDlem ValM-def* $g' \text{ } G' \text{ } g'G'$ **by** *blast*

thus $\langle I, Univ, Univ \rangle, g, G \models^d \varphi$

using *ES*

unfolding *ElementarySubstructure-def*

by (*smt (verit, best) $\gamma(1, \beta) \text{ } G' \text{ } g'$*)

qed
qed

Faithfulness to the standard reading (companion of *Faithful-to-Henkin*).

corollary *Faithful-to-Standard*:

$(\models^{d'} \varphi) = (\forall II \ gg \ GG. \ MinS\text{-}ES\text{-}Univ \ II \ gg \ GG \longrightarrow MinS.ValM$
 $(MinS.DpToShM \ II \ gg \ GG \ \varphi))$
using *Deep'-to-MinS* .

Earlier Range-seeded route to the general-reading converse: the hull packaged with its truth-preservation payoff, derived in one step from *DownwardLowenheimSkolem*.

lemma *ls-hull*:

assumes gD : g into D **and** GE : G into E
obtains $D_0 \ E_0$
where $D_0 \subseteq D$ **and** $E_0 \subseteq E$
and $Range \ g \subseteq D_0$ **and** $Range \ G \subseteq E_0$
and $countable \ \{d. \ D_0 \ d\}$ **and** $countable \ \{S. \ E_0 \ S\}$
and $(\langle I, D_0, E_0 \rangle, g, G \models^d \varphi) = (\langle I, D, E \rangle, g, G \models^d \varphi)$
proof –
have *:
 $Range \ g \subseteq D \ Range \ G \subseteq E$
 $\exists x. \ Range \ g \ x \ \exists x. \ Range \ G \ x$
 $countable \ \{d. \ Range \ g \ d\} \ countable \ \{S. \ Range \ G \ S\}$
using *assms by (auto simp: full-SetCompr-eq)*
obtain $N \ M$
where N : $Range \ g \subseteq N \ countable \ \{d. \ N \ d\}$
 $Range \ G \subseteq M \ countable \ \{S. \ M \ S\}$
and ES : $\langle I, N, M \rangle \subseteq_E \langle I, D, E \rangle$
using *DownwardLowenheimSkolem*
 $[where \ D_0 = Range \ g \ and \ E_0 = Range \ G,$
 $OF \ *(1,2,3,4,5,6), \ of \ I]$
by *metis*
have sub : $N \subseteq D \ M \subseteq E$
and ES' :
 $\bigwedge g \ G \ \varphi. \ G \ into \ M \implies g \ into \ N \implies$
 $(\langle I, D, E \rangle, g, G \models^d \varphi) = (\langle I, N, M \rangle, g, G \models^d \varphi)$
using *ES unfolding ElementarySubstructure-def by auto*
have gn : g into $N \ G$ into M
using *into-Range N(1,3) by (metis (mono-tags, lifting))* +
show *thesis*
using *that[OF sub N(1,3,2,4) ES'[OF gn(2) gn(1), symmetric]]* .
qed

Range reduction: countermodels reduce to range-only countermodels.

lemma *Range-reduction*:

assumes g into D **and** G into E
obtains $g' \ G'$
where $(\langle I, Range \ g', Range \ G' \rangle, g', G' \models^d \varphi)$

$= (\langle I, D, E \rangle, g, G \models^d \varphi)$

proof –

obtain $D_0 \ E_0$

where *:

$\text{Range } g \subseteq D_0 \ \text{Range } G \subseteq E_0$

$\text{countable } \{d. D_0 \ d\} \ \text{countable } \{S. E_0 \ S\}$

and eq :

$(\langle I, D_0, E_0 \rangle, g, G \models^d \varphi) = (\langle I, D, E \rangle, g, G \models^d \varphi)$

using *assms* **by** (*rule ls-hull*)

obtain $g' \ G'$

where $\text{rg}: \text{Range } g' = D_0$ **and** $\text{rG}: \text{Range } G' = E_0$

and coin :

$(\langle I, D_0, E_0 \rangle, g, G \models^d \varphi)$

$= (\langle I, D_0, E_0 \rangle, g', G' \models^d \varphi)$

using *reindex-coincide*[*OF* *(*3,4,1,2*)] **by** *blast*

show *thesis* **using** *that*[*of* $g' \ G'$] $\text{rg} \ \text{rG} \ \text{eq} \ \text{coin}$ **by** *simp*

qed

Range-relative = general (Henkin) deep validity.

theorem *RangeValid-imp-ValD*: $\models^r \varphi \implies \models^d \varphi$

unfolding *RangeValid-def* *ValD-def*

using *Range-reduction* **by** *metis*

corollary *RangeValid-iff-ValD*: $(\models^r \varphi) = (\models^d \varphi)$

using *RangeValid-imp-ValD* *ValD-imp-RangeValid* **by** *blast*

Faithfulness to the general (Henkin) reading. Together with *Faithful-to-Standard* it yields the two faithfulness results; the limitative *Standard-strictly-stronger* below separates them.

corollary *Faithful-to-Henkin*: $(\models^r \varphi) = (\models^d \varphi)$

using *RangeValid-iff-ValD* .

The standard reading is strictly stronger; comprehension witnesses the gap.

lemma *Standard-strictly-stronger*:

$(\forall (\varphi::\mathcal{F}). (\models^d \varphi) \longrightarrow (\models^{d'} \varphi)) \wedge (\exists (\varphi::\mathcal{F}). (\models^{d'} \varphi) \wedge \neg (\models^d \varphi))$

proof

show $\forall (\varphi::\mathcal{F}). (\models^d \varphi) \longrightarrow (\models^{d'} \varphi)$

using *Val* **by** *blast*

next

let $? \varphi = \exists^d_2 (\theta::V2). \forall^d (\theta::V).$

$((\theta::V2)^d((\theta::V)))$

$\longleftrightarrow^d ((\theta::R)^d((\theta::V), (\theta::V)))$

have $\models^{d'} ? \varphi$ **by** (*rule comprehension-atom*)

moreover **have** $\neg (\models^d ? \varphi)$

unfolding *DefD*

apply (*rule notI*,

drule spec[**where** $x=\lambda r \ (a::D) \ b. \ a = b$],

drule spec[**where** $x=Univ::D \Rightarrow bool$],

```

      drule spec[where  $x = \lambda S :: D \Rightarrow \text{bool}. \exists d :: D. \neg S\ d$ ],
      drule spec[where  $x = \lambda x :: V. \text{undefined} :: D$ ],
      drule spec[where  $x = \lambda X :: V\mathbb{2}. \lambda d :: D. \text{False}$ ])
    by (auto simp: SetMod-def EnvMod-def)
  ultimately show  $\exists (\varphi :: \mathcal{F}). (\models^{d'} \varphi) \wedge \neg (\models^d \varphi)$ 
    by blast
qed

end

```

9 Experiments

9.1 Basic landmarks

```

theory MSOinHOL-experiments
  imports
    MSOinHOL-deep
    MSOinHOL-shallow
    MSOinHOL-shallow-minimal
begin

abbreviation (x :: V)  $\equiv 1$ 
abbreviation (y :: V)  $\equiv 2$ 
abbreviation (z :: V)  $\equiv 3$ 
abbreviation (X :: V $\mathbb{2}$ )  $\equiv 1$ 
abbreviation (Y :: V $\mathbb{2}$ )  $\equiv 2$ 

consts P :: R

```

9.1.1 Propositional tautologies, lifted to MSO in all three embeddings

```

lemma  $\models^d (P^d(x, x) \supset^d P^d(x, x))$ 
  unfolding DefD by simp

lemma  $\models^s (P^s(x, x) \supset^s P^s(x, x))$ 
  unfolding DefS by simp

lemma  $\models^m (P^m(x, x) \supset^m P^m(x, x))$ 
  unfolding DefM by simp

lemma  $\models^d (\neg^d \neg^d P^d(x, y)) \supset^d P^d(x, y)$ 
  unfolding DefD by auto

lemma  $\models^s (\neg^s \neg^s P^s(x, y)) \supset^s P^s(x, y)$ 
  unfolding DefS by auto

lemma  $\models^m (\neg^m \neg^m P^m(x, y)) \supset^m P^m(x, y)$ 
  unfolding DefM by auto

```

9.1.2 First-order tautologies

The individual domain is inhabited whenever some assignment exists.

lemma $\models^d (\forall^d x. P^d(x, x)) \supset^d (\exists^d x. P^d(x, x))$
unfolding *DefD* by *auto*

lemma $\models^s (\forall^s x. P^s(x, x)) \supset^s (\exists^s x. P^s(x, x))$
unfolding *DefS* by *auto*

lemma $\models^m (\forall^m x. P^m(x, x)) \supset^m (\exists^m x. P^m(x, x))$
unfolding *DefM* by *auto*

9.1.3 Membership tautology

Every individual that is in X is in X .

lemma $\models^d (\forall^d x. (X^d(x) \supset^d X^d(x)))$
unfolding *DefD* by *auto*

lemma $\models^s (\forall^s x. (X^s(x) \supset^s X^s(x)))$
unfolding *DefS* by *auto*

lemma $\models^m (\forall^m x. (X^m(x) \supset^m X^m(x)))$
unfolding *DefM* by *auto*

9.1.4 Monadic comprehension

Headline second-order validity: the set of P -self-related individuals exists. Stated over the full second-order domain, where every monadic set is admissible.

lemma *comprehension-d*:
 $\models^{d'} (\exists^d {}_2 X. \forall^d x. ((X^d(x)) \longleftrightarrow^d P^d(x, x)))$
unfolding *DefD* by (*intro allI*; *simp*) *meson*

lemma *comprehension-s*:
 $\models^{s'} (\exists^s {}_2 X. \forall^s x. ((X^s(x)) \longleftrightarrow^s P^s(x, x)))$
unfolding *DefS* by (*intro allI*; *simp*) *meson*

A universal monadic set exists over the full domain (the predicate that holds everywhere).

lemma $\models^{d'} (\exists^d {}_2 X. \forall^d x. (X^d(x)))$
unfolding *DefD* by (*simp*, *rule exI*[**where** $x = \lambda d. \text{True}$]) *simp*

9.1.5 Invalid schemata

nitpick reports finite countermodels (using the full-domain relation).

Not every monadic set is inhabited: the empty set is a countermodel to $\forall X. \exists x. x \in X$.

lemma $\models^{d'}$ $(\forall^d_2 X. \exists^d x. (X^d(x)))$
unfolding *DefD apply simp nitpick oops*

Symmetry of P is not implied: *nitpick* produces a 2-element counter-model.

lemma $\models^{d'}$ $(\forall^d x. \forall^d y. (P^d(x,y) \supset^d P^d(y,x)))$
unfolding *DefD apply simp nitpick oops*

No monadic set can contain and exclude every individual at once: $\exists X. \forall x. x \in X \wedge \neg x \in X$ is unsatisfiable, hence invalid.

lemma $\models^{d'}$ $(\exists^d_2 X. \forall^d x. ((X^d(x)) \wedge^d \neg^d (X^d(x))))$
unfolding *DefD apply simp nitpick oops*

end

9.2 Classical MSO landmarks

theory *MSOinHOL-experiments-classic*
imports
MSOinHOL-deep
MSOinHOL-shallow
MSOinHOL-shallow-minimal
begin

abbreviation $(x::V) \equiv 1$
abbreviation $(y::V) \equiv 2$
abbreviation $(z::V) \equiv 3$
abbreviation $(u::V) \equiv 4$
abbreviation $(v::V) \equiv 5$
abbreviation $(X::V\mathcal{2}) \equiv 1$
abbreviation $(Y::V\mathcal{2}) \equiv 2$
abbreviation $(Z::V\mathcal{2}) \equiv 3$

consts $P :: R$

9.2.1 Boolean closure (Büchi 1960; Thomas 1997) under $\models^{d'}$

lemma *complement-d*:
 $\models^{d'} (\forall^d_2 X. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d \neg^d X^d(x)))$
unfolding *DefD*
apply *(intro allI, simp add: SetMod-def EnvMod-def, intro allI)*
subgoal for S
by *(rule exI[of - $\lambda d. \neg S d$]) (auto simp: SetMod-def EnvMod-def)*
done

lemma *intersection-d*:
 $\models^{d'} (\forall^d_2 X. \forall^d_2 Y. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \wedge^d Y^d(x))))$
unfolding *DefD*
apply *(intro allI, simp add: SetMod-def EnvMod-def, intro allI)*

subgoal for $S Sa$
 by (rule exI[of - $\lambda d. S d \wedge Sa d$])
 (auto simp: SetMod-def EnvMod-def)
done

lemma union-d:
 $\models^{d'} (\forall^d_2 X. \forall^d_2 Y. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \vee^d Y^d(x))))$
unfolding DefD
apply (intro allI, simp add: SetMod-def EnvMod-def, intro allI)
subgoal for $S Sa$
 by (rule exI[of - $\lambda d. S d \vee Sa d$])
 (auto simp: SetMod-def EnvMod-def)
done

9.2.2 Graph operations (Courcelle 2012)

lemma separation-d:
 $\models^{d'} (\forall^d_2 X. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \wedge^d P^d(x,x))))$
unfolding DefD
apply (intro allI, simp add: SetMod-def EnvMod-def, intro allI)
subgoal for $I S$
 by (rule exI[of - $\lambda d. S d \wedge I P d d$])
 (auto simp: SetMod-def EnvMod-def)
done

lemma image-d:
 $\models^{d'} (\forall^d_2 X. \exists^d_2 Y. \forall^d x. (Y^d(x) \longleftrightarrow^d \exists^d y. (X^d(y) \wedge^d P^d(y,x))))$
unfolding DefD
apply (intro allI, simp add: SetMod-def EnvMod-def, intro allI)
subgoal for $I S$
 by (rule exI[of - $\lambda d. \exists d'. S d' \wedge I P d' d$])
 (auto simp: SetMod-def EnvMod-def)
done

lemma preimage-d:
 $\models^{d'} (\forall^d_2 X. \exists^d_2 Y. \forall^d x. (Y^d(x) \longleftrightarrow^d \exists^d y. (P^d(x,y) \wedge^d X^d(y))))$
unfolding DefD
apply (intro allI, simp add: SetMod-def EnvMod-def, intro allI)
subgoal for $I S$
 by (rule exI[of - $\lambda d. \exists d'. I P d d' \wedge S d'$])
 (auto simp: SetMod-def EnvMod-def)
done

Reachability (Basin and Klarlund 1995): not universally valid; reflexive variant is.

lemma reachability-not-valid-d:
 $\models^{d'} (\forall^d_2 Z. ((Z^d(x) \wedge^d (\forall^d u. (Z^d(u) \supset^d \forall^d v. (P^d(u,v) \supset^d Z^d(v)))) \supset^d Z^d(y)))$
unfolding DefD apply simp nitpick oops

lemma *reachability-reflexive-d*:

$\models^{d'} (\forall^d_2 Z. ((Z^d(x) \wedge^d (\forall^d u. (Z^d(u) \supset^d \forall^d v. (P^d(u,v) \supset^d Z^d(v)))) \supset^d Z^d(x)))$

unfolding *DefD* **by** *simp*

2-colorability (Thomas 1997): refuted on the triangle K_3 (the complete graph on three vertices).

lemma *two-colorability-not-valid-d*:

$\models^{d'} (\exists^d_2 Z. \forall^d x. \forall^d y. (P^d(x,y) \supset^d (Z^d(x) \longleftrightarrow^d \neg^d Z^d(y))))$

unfolding *DefD* **apply** *simp* **nitpick** **oops**

9.2.3 Maximal-shallow embedding

Same landmarks in the maximal-shallow embedding: structurally identical proofs.

lemma *complement-s*:

$\models^{s'} (\forall^s_2 X. \exists^s_2 Z. \forall^s x. (Z^s(x) \longleftrightarrow^s \neg^s X^s(x)))$

unfolding *DefS*

apply (*intro allI*, *simp*, *intro allI*)

subgoal for S **by** (*rule exI*[*of* - $\lambda d. \neg S d$]) *auto*

done

lemma *intersection-s*:

$\models^{s'} (\forall^s_2 X. \forall^s_2 Y. \exists^s_2 Z. \forall^s x. (Z^s(x) \longleftrightarrow^s (X^s(x) \wedge^s Y^s(x))))$

unfolding *DefS*

apply (*intro allI*, *simp*, *intro allI*)

subgoal for S Sa **by** (*rule exI*[*of* - $\lambda d. S d \wedge Sa d$]) *auto*

done

lemma *union-s*:

$\models^{s'} (\forall^s_2 X. \forall^s_2 Y. \exists^s_2 Z. \forall^s x. (Z^s(x) \longleftrightarrow^s (X^s(x) \vee^s Y^s(x))))$

unfolding *DefS*

apply (*intro allI*, *simp*, *intro allI*)

subgoal for S Sa **by** (*rule exI*[*of* - $\lambda d. S d \vee Sa d$]) *auto*

done

lemma *separation-s*:

$\models^{s'} (\forall^s_2 X. \exists^s_2 Z. \forall^s x. (Z^s(x) \longleftrightarrow^s (X^s(x) \wedge^s P^s(x,x))))$

unfolding *DefS*

apply (*intro allI*, *simp*, *intro allI*)

subgoal for I S **by** (*rule exI*[*of* - $\lambda d. S d \wedge I P d d$]) *auto*

done

lemma *image-s*:

$\models^{s'} (\forall^s_2 X. \exists^s_2 Y. \forall^s x. (Y^s(x) \longleftrightarrow^s (\exists^s y. (X^s(y) \wedge^s P^s(y,x))))$

unfolding *DefS*

apply (*intro allI*, *simp*, *intro allI*)

subgoal for I S

by (*rule exI*[*of* - $\lambda d. \exists d'. S d' \wedge I P d' d$]) *auto*

done

lemma *preimage-s*:

$\models^{s'} (\forall^{s_2} X. \exists^{s_2} Y. \forall^s x. (Y^s(x) \longleftrightarrow^s (\exists^s y. (P^s(x,y) \wedge^s X^s(y)))))$
unfolding *DefS*
apply (*intro allI*, *simp*, *intro allI*)
subgoal for *I S*
by (*rule exI*[*of* - $\lambda d. \exists d'. I P d d' \wedge S d'$]) *auto*
done

lemma *reachability-not-valid-s*:

$\models^{s'} (\forall^{s_2} Z. ((Z^s(x) \wedge^s (\forall^s u. (Z^s(u) \supset^s (\forall^s v. (P^s(u,v) \supset^s Z^s(v))))) \supset^s Z^s(y)))$
unfolding *DefS* **apply** (*intro allI*) **apply** *simp nitpick oops*

lemma *reachability-reflexive-s*:

$\models^{s'} (\forall^{s_2} Z. ((Z^s(x) \wedge^s (\forall^s u. (Z^s(u) \supset^s (\forall^s v. (P^s(u,v) \supset^s Z^s(v))))) \supset^s Z^s(x)))$
unfolding *DefS* **by** *simp*

lemma *two-colorability-not-valid-s*:

$\models^{s'} (\exists^{s_2} Z. \forall^s x. \forall^s y. (P^s(x,y) \supset^s (Z^s(x) \longleftrightarrow^s \neg^s Z^s(y))))$
unfolding *DefS* **apply** (*intro allI*) **apply** *simp nitpick oops*

9.2.4 Minimal-shallow embedding

Minimal embedding: SO quantifier ranges over the countable *Range GG* (via *nat*), not all of *Pow(D)*. Nitpick can only certify a POTENTIAL counter-model.

lemma *complement-m-not-valid*:

$\models^m (\forall^{m_2} X. \exists^{m_2} Z. \forall^m x. (Z^m(x) \longleftrightarrow^m \neg^m X^m(x)))$
unfolding *DefM* **nitpick**[*expect=potential*] **oops**

lemma *intersection-m-not-valid*:

$\models^m (\forall^{m_2} X. \forall^{m_2} Y. \exists^{m_2} Z. \forall^m x. (Z^m(x) \longleftrightarrow^m (X^m(x) \wedge^m Y^m(x))))$
unfolding *DefM* **nitpick**[*expect=potential*] **oops**

lemma *reachability-not-valid-m*:

$\models^m (\forall^{m_2} Z. ((Z^m(x) \wedge^m (\forall^m u. (Z^m(u) \supset^m (\forall^m v. (P^m(u,v) \supset^m Z^m(v))))) \supset^m Z^m(y)))$
unfolding *DefM* **nitpick**[*expect=potential*] **oops**

Reflexive reachability: the conclusion is the first conjunct of the antecedent; genuinely valid.

lemma *reachability-reflexive-m*:

$\models^m (\forall^{m_2} Z. ((Z^m(x) \wedge^m (\forall^m u. (Z^m(u) \supset^m (\forall^m v. (P^m(u,v) \supset^m Z^m(v))))) \supset^m Z^m(x)))$
unfolding *DefM* **by** *simp*

lemma *two-colorability-not-valid-m*:

$\models^m (\exists^{m_2} Z. \forall^m x. \forall^m y. (P^m(x,y) \supset^m (Z^m(x) \longleftrightarrow^m \neg^m Z^m(y))))$
unfolding *DefM* **nitpick**[*expect=potential*] **oops**

end

9.3 Locale-based variants

theory *MSOinHOL-experiments-locale*
imports *MSOinHOL-faithfulness-locale*
begin

Demonstration: minimal-shallow proofs that hold in every interpretation of the *MinS* locale transfer to validity in the model class whose domains are the assignment ranges.

Derived rule: from validity in every minimal interpretation to range-relative deep validity.

lemmas *MinS-to-Deep* = *FaithfulMS-all*[*THEN iffD1, rule-format*]

Cosmetic locale-level rewrite rules for derived connectives under *Dp-ToShM*.

lemma (in *MinS*) *OrM-simp* [*DefM*]: $\llbracket \varphi \vee^d \psi \rrbracket = \llbracket \varphi \rrbracket \vee^m \llbracket \psi \rrbracket$
by (*simp add: OrD-def OrM-def*)

lemma (in *MinS*) *ImpM-simp* [*DefM*]: $\llbracket \varphi \supset^d \psi \rrbracket = \llbracket \varphi \rrbracket \supset^m \llbracket \psi \rrbracket$
by (*simp add: ImpD-def DefM*)

Example: a membership tautology proved using only minimal-shallow infrastructure.

lemma (in *MinS*) *Mem-MinS*: $\models^m \llbracket (P^d(x,x)) \supset^d (P^d(x,x)) \rrbracket$
by (*auto simp: DefM*)

Transfer to the range-relative deep embedding requires no further work.

lemma $\langle I, \text{Range } g, \text{Range } G \rangle, g, G \models^d ((P^d(x,x)) \supset^d (P^d(x,x)))$
by (*rule MinS-to-Deep*) (*blast intro: MinS.Mem-MinS*)

end

9.4 Elementary minimal embedding (setup)

theory *MSOinHOL-shallow-minimal-elementary*
imports
MSOinHOL-faithfulness-locale
MSOinHOL-lowenheim-skolem
begin

Extra simp rules for *DpToShS* (derived quantifiers and connectives).

lemma (in *MinS*) *DpToShS-All* [*simp*]:
 $\llbracket \forall^d x. \varphi \rrbracket = (\forall^m d. \llbracket [x \leftarrow_r d](\varphi) \rrbracket)$
unfolding *DefD DefM* **by** (*simp add: ExM-def NegM-def ren-subst-def*)

lemma (in *MinS*) *DpToShS-All2* [simp]:
 $(\forall^d x. \varphi) = (\forall^{m_2} d. (\llbracket x \leftarrow_{r_2} d \rrbracket (\varphi)))$
unfolding *DefD DefM* **using** *MinS.FaithfulMDlem* **by** *auto*

lemma (in *MinS*) *DpToShS-Equiv* [simp]:
 $(\varphi \longleftrightarrow^d \psi) = ((\varphi) \longleftrightarrow^m (\psi))$
unfolding *DefD DefM* **by** (*simp add: AndM-def NegM-def*)

lemma (in *MinS*) *DpToShS-Imp* [simp]:
 $(\varphi \supset^d \psi) = ((\varphi) \supset^m (\psi))$
unfolding *DefD DefM* **by** (*simp add: AndM-def NegM-def*)

Elementary constants-only presentation: *II*, *gg*, *GG* chosen so that the global interpretation is a *MinS-ES-Univ* (existence via *Deep'-to-MinS*).

consts *II* :: $\mathcal{I}$ *gg* :: $\mathcal{E}$ *GG* :: $\mathcal{G}$

specification (*II gg GG*) *ES-Univ: MinS-ES-Univ II gg GG*
by (*meson Deep'-to-MinS RelativeTruthD.simps ValD'-def*)

global-interpretation *MinS-ES-Univ II gg GG*
using *ES-Univ* **by** *auto*

notation *AtmM* $(\neg^m (-))$
and *PrdM* $(\neg^m (-))$
and *NegM* $(\neg^m - [58] 59)$
and *AndM* $(\text{infixr } \wedge^m 56)$
and *ExM* $(\text{binder } \exists^m 53)$
and *ExM2* $(\text{binder } \exists^{m_2} 53)$
and *OrM* $(\text{infixr } \vee^m 54)$
and *ImpM* $(\text{infixr } \supset^m 55)$
and *IffM* $(\text{infixr } \longleftrightarrow^m 54)$
and *AllM* $(\text{binder } \forall^m 53)$
and *AllM2* $(\text{binder } \forall^{m_2} 53)$
and *ValM* $(\models^m - 9)$
and *DpToShM* $(\llbracket - \rrbracket)$

Faithfulness under universal carriers.

lemma *FaithfulMD-ES*: $(\models^m (\varphi)) = \langle II, Univ, Univ \rangle, gg, GG \models^d \varphi$
using *FaithfulMD N-valid-ES-Univ* **by** *blast*

Inheritance: standard validity transfers to minimal validity.

lemma *ValidM-if-ValidD'*: $(\models^{d'} \varphi) \implies (\models^m (\varphi))$
by (*metis MinS-ES-Univ-axioms Deep'-to-MinS*)

end

9.5 Elementary minimal embedding (landmarks)

theory *MSOinHOL-experiments-classic-elementary*

imports
MSOinHOL-deep
MSOinHOL-shallow
MSOinHOL-shallow-minimal-elementary
begin

Extra simp rules for derived quantifiers / connectives.

lemma *ren-for-subst2-simp-All* [*simp*]:
 $\text{ren-for-subst2 } X \ Z \ (\forall^d_2 Y. \ \varphi) =$
 (if $Y = Z \wedge X \text{ free2-in } \varphi$
 then let $f = \max (\text{fresh2 } \varphi) \ (Z+1)$; $\varphi' = [Y \leftarrow_2 f](\varphi)$
 in $(\forall^d_2 f. \text{ren-for-subst2 } X \ Z \ \varphi')$
 else $\forall^d_2 Y. \text{ren-for-subst2 } X \ Z \ \varphi$)
unfolding *DefD* **by** (*auto simp: Let-def*)

lemma *subst2-all* [*simp*]:
 $[X \leftarrow_2 Z](\forall^d_2 Y. \ \varphi) =$
 (if $X = Y$ then $(\forall^d_2 Y. \ \varphi)$ else $(\forall^d_2 Y. [X \leftarrow_2 Z](\varphi))$)
unfolding *DefD* **by** (*auto simp: Let-def*)

lemma *free2-in-equiv* [*simp*]:
 $X \text{ free2-in } (\varphi \longleftrightarrow^d \psi) = (X \text{ free2-in } \varphi \vee X \text{ free2-in } \psi)$
by (*auto simp add: DefD*)

lemma *free2-in-all* [*simp*]:
 $X \text{ free2-in } (\forall^d y. \ \varphi) = (X \text{ free2-in } \varphi)$
by (*simp add: AllD-def*)

lemma *free2-in-all2* [*simp*]:
 $X \text{ free2-in } (\forall^d_2 Y. \ \varphi) = (X \text{ free2-in } \varphi \wedge X \neq Y)$
by (*metis AllD2-def is-free2.simps*)

Some abbreviations for variables.

abbreviation $(x::V) \equiv 1$
abbreviation $(y::V) \equiv 2$
abbreviation $(z::V) \equiv 3$
abbreviation $(u::V) \equiv 4$
abbreviation $(v::V) \equiv 5$
abbreviation $(X::V2) \equiv 1$
abbreviation $(Y::V2) \equiv 2$
abbreviation $(Z::V2) \equiv 3$

consts $P :: R$

9.5.1 Boolean closure (Büchi 1960; Thomas 1997)

Under $\models^{d'}$: witnesses supplied via *exI*.

lemma *complement-d*:
 $\models^{d'} (\forall^d_2 X. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d \neg^d X^d(x)))$

unfolding *DefD*
apply (*intro allI*, *simp add: SetMod-def EnvMod-def*, *intro allI*)
subgoal for *S*
by (*rule exI[of - $\lambda d. \neg S d$]*) (*auto simp: SetMod-def EnvMod-def*)
done

lemma *intersection-d*:
 $\models^{d'} (\forall^d_2 X. \forall^d_2 Y. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \wedge^d Y^d(x))))$
unfolding *DefD*
apply (*intro allI*, *simp add: SetMod-def EnvMod-def*, *intro allI*)
subgoal for *S Sa*
by (*rule exI[of - $\lambda d. S d \wedge Sa d$]*)
(*auto simp: SetMod-def EnvMod-def*)
done

lemma *intersection-d'*:
 $\models^{d'} (\exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \wedge^d Y^d(x))))$
by (*simp add: comprehension-schema*)

lemma *union-d*:
 $\models^{d'} (\forall^d_2 X. \forall^d_2 Y. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \vee^d Y^d(x))))$
unfolding *DefD*
apply (*intro allI*, *simp add: SetMod-def EnvMod-def*, *intro allI*)
subgoal for *S Sa*
by (*rule exI[of - $\lambda d. S d \vee Sa d$]*)
(*auto simp: SetMod-def EnvMod-def*)
done

9.5.2 Graph operations (Courcelle 2012)

lemma *separation-d*:
 $\models^{d'} (\forall^d_2 X. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \wedge^d P^d(x,x))))$
unfolding *DefD*
apply (*intro allI*, *simp add: SetMod-def EnvMod-def*, *intro allI*)
subgoal for *I S*
by (*rule exI[of - $\lambda d. S d \wedge I P d d$]*)
(*auto simp: SetMod-def EnvMod-def*)
done

lemma *image-d*:
 $\models^{d'} (\forall^d_2 X. \exists^d_2 Y. \forall^d x. (Y^d(x) \longleftrightarrow^d \exists^d y. (X^d(y) \wedge^d P^d(y,x))))$
unfolding *DefD*
apply (*intro allI*, *simp add: SetMod-def EnvMod-def*, *intro allI*)
subgoal for *I S*
by (*rule exI[of - $\lambda d. \exists d'. S d' \wedge I P d' d$]*)
(*auto simp: SetMod-def EnvMod-def*)
done

lemma *preimage-d*:

$\models^{d'} (\forall^d X. \exists^d Y. \forall^d x. (Y^d(x) \longleftrightarrow^d \exists^d y. (P^d(x,y) \wedge^d X^d(y))))$
unfolding *DefD*
apply (*intro allI*, *simp add: SetMod-def EnvMod-def*, *intro allI*)
subgoal for *I S*
by (*rule exI[of - $\lambda d. \exists d'. I P d d' \wedge S d$]*)
(auto simp: SetMod-def EnvMod-def)
done

Reachability (Basin and Klarlund 1995): not universally valid; reflexive variant is.

lemma *reachability-not-valid-d:*

$\models^{d'} (\forall^d Z. ((Z^d(x) \wedge^d (\forall^d u. (Z^d(u) \supset^d \forall^d v. (P^d(u,v) \supset^d Z^d(v)))) \supset^d Z^d(y)))$
unfolding *DefD* **apply** *simp nitpick oops*

lemma *reachability-reflexive-d:*

$\models^{d'} (\forall^d Z. ((Z^d(x) \wedge^d (\forall^d u. (Z^d(u) \supset^d \forall^d v. (P^d(u,v) \supset^d Z^d(v)))) \supset^d Z^d(x)))$
unfolding *DefD* **by** *simp*

2-colorability (Thomas 1997): refuted on the triangle K_3 (the complete graph on three vertices).

lemma *two-colorability-not-valid-d:*

$\models^{d'} (\exists^d Z. \forall^d x. \forall^d y. (P^d(x,y) \supset^d (Z^d(x) \longleftrightarrow^d \neg^d Z^d(y))))$
unfolding *DefD* **apply** *simp nitpick oops*

9.5.3 Maximal-shallow embedding

Same landmarks in the maximal-shallow embedding: structurally identical proofs.

lemma *complement-s:*

$\models^{s'} (\forall^s X. \exists^s Z. \forall^s x. (Z^s(x) \longleftrightarrow^s \neg^s X^s(x)))$
unfolding *DefS*
apply (*intro allI*, *simp*, *intro allI*)
subgoal for *S* **by** (*rule exI[of - $\lambda d. \neg S d$]*) *auto*
done

lemma *intersection-s:*

$\models^{s'} (\forall^s X. \forall^s Y. \exists^s Z. \forall^s x. (Z^s(x) \longleftrightarrow^s (X^s(x) \wedge^s Y^s(x))))$
unfolding *DefS*
apply (*intro allI*, *simp*, *intro allI*)
subgoal for *S Sa* **by** (*rule exI[of - $\lambda d. S d \wedge Sa d$]*) *auto*
done

lemma *union-s:*

$\models^{s'} (\forall^s X. \forall^s Y. \exists^s Z. \forall^s x. (Z^s(x) \longleftrightarrow^s (X^s(x) \vee^s Y^s(x))))$
unfolding *DefS*
apply (*intro allI*, *simp*, *intro allI*)
subgoal for *S Sa* **by** (*rule exI[of - $\lambda d. S d \vee Sa d$]*) *auto*
done

lemma separation-s:

$\models^{s'} (\forall^s_2 X. \exists^s_2 Z. \forall^s x. (Z^s(x) \longleftrightarrow^s (X^s(x) \wedge^s P^s(x,x))))$
unfolding *DefS*
apply (*intro allI, simp, intro allI*)
subgoal for *I S* **by** (*rule exI[of - $\lambda d. S d \wedge I P d d$] auto*)
done

lemma image-s:

$\models^{s'} (\forall^s_2 X. \exists^s_2 Y. \forall^s x. (Y^s(x) \longleftrightarrow^s (\exists^s y. (X^s(y) \wedge^s P^s(y,x))))$
unfolding *DefS*
apply (*intro allI, simp, intro allI*)
subgoal for *I S*
by (*rule exI[of - $\lambda d. \exists d'. S d' \wedge I P d' d$] auto*)
done

lemma preimage-s:

$\models^{s'} (\forall^s_2 X. \exists^s_2 Y. \forall^s x. (Y^s(x) \longleftrightarrow^s (\exists^s y. (P^s(x,y) \wedge^s X^s(y))))$
unfolding *DefS*
apply (*intro allI, simp, intro allI*)
subgoal for *I S*
by (*rule exI[of - $\lambda d. \exists d'. I P d d' \wedge S d'$] auto*)
done

lemma reachability-not-valid-s:

$\models^{s'} (\forall^s_2 Z. ((Z^s(x) \wedge^s (\forall^s u. (Z^s(u) \supset^s (\forall^s v. (P^s(u,v) \supset^s Z^s(v)))))) \supset^s Z^s(y)))$
unfolding *DefS* **apply** (*intro allI*) **apply** *simp nitpick oops*

lemma reachability-reflexive-s:

$\models^{s'} (\forall^s_2 Z. ((Z^s(x) \wedge^s (\forall^s u. (Z^s(u) \supset^s (\forall^s v. (P^s(u,v) \supset^s Z^s(v)))))) \supset^s Z^s(x)))$
unfolding *DefS* **by** *simp*

lemma two-colorability-not-valid-s:

$\models^{s'} (\exists^s_2 Z. \forall^s x. \forall^s y. (P^s(x,y) \supset^s (Z^s(x) \longleftrightarrow^s \neg^s Z^s(y))))$
unfolding *DefS* **apply** (*intro allI*) **apply** *simp nitpick oops*

9.5.4 Transfer to the elementary minimal embedding

Via the *DpToShS* normal-form and *ValidM-if-ValidD'*.

lemma compl-m-DpToSh:

$(\forall^d_2 X. \exists^d_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d \neg^d X^d(x))) = (\forall^m_2 X. \exists^m_2 Z. \forall^m x. (Z^m(x) \longleftrightarrow^m \neg^m X^m(x)))$
by (*auto simp add: ren-subst2-def ren-subst-def Let-def*)
(auto simp: DefD DefM ren-subst-def)

lemma compl-m-valid:

$\models^m (\forall^m_2 X. \exists^m_2 Z. \forall^m x. (Z^m(x) \longleftrightarrow^m \neg^m X^m(x)))$
using *compl-m-DpToSh ValidM-if-ValidD'[OF complement-d]* **by** *simp*

lemma inters-m-DpToSh':

$$\llbracket \exists^d {}_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \wedge^d Y^d(x))) \rrbracket = (\exists {}^m {}_2 Z. \forall^m x. (Z^m(x) \longleftrightarrow^m (X^m(x) \wedge^m Y^m(x))))$$
by (*auto simp add: ren-subst2-def ren-subst-def Let-def*)
(auto simp: DefD DefM ren-subst-def)

lemma *DpToSh-allegI*:

$$(\bigwedge d. \llbracket [X \leftarrow_{r2} d](\varphi) \rrbracket = \psi \ d) \implies (\llbracket \forall^d {}_2 X. \varphi \rrbracket = (\forall {}^m {}_2 X. \psi \ X))$$
by *simp*

lemma *all-m-eqI*:

$$(\bigwedge X. \varphi \ X = \psi \ X) \implies (\forall {}^m {}_2 X. \varphi \ X) = (\forall {}^m {}_2 X. \psi \ X)$$
by *presburger*

lemmas *DpToSh-simps = DefM Let-def ren-subst-def DefD ren-subst2-def*

lemma *intersect-m-DpToSh*:

$$\llbracket \forall^d {}_2 X. \forall^d {}_2 Y. \exists^d {}_2 Z. \forall^d x. (Z^d(x) \longleftrightarrow^d (X^d(x) \wedge^d Y^d(x))) \rrbracket = (\forall {}^m {}_2 X. \forall {}^m {}_2 Y. \exists {}^m {}_2 Z. \forall^m x. (Z^m(x) \longleftrightarrow^m (X^m(x) \wedge^m Y^m(x))))$$

— Push *DpToShM* through the two outer set quantifiers and split on whether the fresh index d chosen by the renaming coincides with Z , with Y , or with neither. Each case is discharged inside its own *subgoal* block, so no *auto* step silently hands its leftover goals to the next one.

apply (*rule DpToSh-allegI*,
simp only: ren-subst2-def subst2-all ren-for-subst2-simp-All,
simp)

subgoal for d

apply (*cases* $d = Z$, *simp*, *rule all-m-eqI*)

subgoal by (*auto simp: DpToSh-simps*)

apply (*simp*, *cases* $d = Y$, *simp*)

subgoal by (*auto simp: DpToSh-simps; metis*)

— Remaining case $d \notin \{Y, Z\}$: normalise the renamed body and offer the bound predicate D as the existential witness. The fresh index $\max Z$ ($\max d \ Y$) exceeds both Y and d , so the two guarded substitutions collapse. Normalisation and the witness step are kept together in one *subgoal*, so any change in the shape of the *auto*-normalised goal fails here rather than downstream.

apply (*simp*, *rule all-m-eqI*)

subgoal for da

apply (*auto simp: ren-subst2-def ren-subst-def Let-def*)

subgoal apply (*auto simp: DpToSh-simps*)

subgoal for D

apply (*rule exI[where* $x=D$ *]*)

apply (*subgoal-tac Suc* ($\max Z$ ($\max d \ Y$)) $\neq d$ $\max Z$ ($\max d \ Y$) $\neq Y$)

apply *simp*

apply *simp*

apply (*simp add: le-imp-less-Suc max.coboundedI2*)

done

done

by (*auto simp: DpToSh-simps*)

done

done

lemma *intersection-m-valid:*

$\models^m (\forall^m {}_2 X. \forall^m {}_2 Y. \exists^m {}_2 Z. \forall^m x. (Z^m(x) \longleftrightarrow^m (X^m(x) \wedge^m Y^m(x))))$
using *ValM-def ValidM-if-ValidD' intersection-d intersect-m-DpToSh*
by *blast*

Reachability and 2-colorability remain refuted in the elementary minimal embedding.

lemma *reachability-not-valid-m:*

$\models^m (\forall^m {}_2 Z. ((Z^m(x) \wedge^m (\forall^m u. (Z^m(u) \supset^m (\forall^m v. (P^m(u,v) \supset^m Z^m(v)))))) \supset^m Z^m(y)))$
unfolding *DefM* **oops**

lemma *reachability-reflexive-m:*

$\models^m (\forall^m {}_2 Z. ((Z^m(x) \wedge^m (\forall^m u. (Z^m(u) \supset^m (\forall^m v. (P^m(u,v) \supset^m Z^m(v)))))) \supset^m Z^m(x)))$
unfolding *DefM* **by** *simp*

lemma *two-colorability-not-valid-m:*

$\models^m (\exists^m {}_2 Z. \forall^m x. \forall^m y. (P^m(x,y) \supset^m (Z^m(x) \longleftrightarrow^m \neg^m Z^m(y))))$
unfolding *DefM* **oops**

end

9.6 Additional landmarks

theory *MSOinHOL-experiments-extra*

imports

MSOinHOL-deep

MSOinHOL-shallow

MSOinHOL-shallow-minimal

begin

Additional MSO landmarks complementing the experiments in *MSOinHOL-experiments-classic*: 3-colorability, vertex cover, transitivity, symmetry and triangle existence, each replicated across the deep, maximal-shallow and minimal-shallow embeddings.

abbreviation $(x::V) \equiv 1$

abbreviation $(y::V) \equiv 2$

abbreviation $(z::V) \equiv 3$

abbreviation $(u::V) \equiv 4$

abbreviation $(v::V) \equiv 5$

abbreviation $(X::V2) \equiv 1$

abbreviation $(Y::V2) \equiv 2$

abbreviation $(Z::V2) \equiv 3$

consts $P :: R$

9.6.1 Deep embedding (under $\models^{d'}$)

3-colorability (Thomas 1997): refuted on K_4 (the complete graph on four vertices, which needs four colors). Generalisation of *two-colorability-not-valid-d*.

lemma *three-colorability-not-valid-d*:

$$\models^{d'} (\exists^d X. \exists^d Y. \exists^d Z. ((\forall^d u. (X^d(u) \vee^d Y^d(u) \vee^d Z^d(u))) \wedge^d (\forall^d u. \forall^d v. (P^d(u,v) \supset^d ((\neg^d (X^d(u) \wedge^d X^d(v))) \wedge^d (\neg^d (Y^d(u) \wedge^d Y^d(v))) \wedge^d (\neg^d (Z^d(u) \wedge^d Z^d(v))))))))))$$

unfolding *DefD* **apply** *simp* **nitpick** **oops**

Vertex cover existence: the predicate *Univ* is a witness, so the schema is universally valid.

lemma *vertex-cover-valid-d*:

$$\models^{d'} (\exists^d X. \forall^d u. \forall^d v. (P^d(u,v) \supset^d (X^d(u) \vee^d X^d(v))))$$

unfolding *DefD*

apply (*intro allI*, *simp add: SetMod-def EnvMod-def*)

by (*rule exI[of - Univ]*) (*auto simp: SetMod-def EnvMod-def*)

Transitivity of *P*: not universally valid; refuted by any 3-path $a \rightarrow b \rightarrow c$ without the closing edge $a \rightarrow c$.

lemma *transitivity-not-valid-d*:

$$\models^{d'} (\forall^d x. \forall^d y. \forall^d z. (P^d(x,y) \supset^d (P^d(y,z) \supset^d P^d(x,z))))$$

unfolding *DefD* **apply** *simp* **nitpick** **oops**

Symmetric extension: stating *P* is symmetric is also not universally valid; refuted by any directed edge without its converse.

lemma *symmetry-not-valid-d*:

$$\models^{d'} (\forall^d x. \forall^d y. (P^d(x,y) \supset^d P^d(y,x)))$$

unfolding *DefD* **apply** *simp* **nitpick** **oops**

Existence of a triangle: not universally valid; refuted on any triangle-free graph (e.g. the empty graph).

lemma *triangle-exists-not-valid-d*:

$$\models^{d'} (\exists^d x. \exists^d y. \exists^d z. (P^d(x,y) \wedge^d P^d(y,z) \wedge^d P^d(x,z)))$$

unfolding *DefD* **apply** *simp* **nitpick** **oops**

Existence of an edge: refuted on the empty graph.

lemma *has-edge-not-valid-d*:

$$\models^{d'} (\exists^d x. \exists^d y. P^d(x,y))$$

unfolding *DefD* **apply** *simp* **nitpick** **oops**

Loop-freeness: stating *P* has no self-loops is refuted by any graph with a fixed point.

lemma *loop-free-not-valid-d*:

$$\models^{d'} (\forall^d x. \neg^d P^d(x,x))$$

unfolding *DefD* **apply** *simp* **nitpick** **oops**

9.6.2 Maximal-shallow embedding (under $\models^s$)

lemma *three-colorability-not-valid-s:*

$\models^{s'} (\exists^s_2 X. \exists^s_2 Y. \exists^s_2 Z. ((\forall^s u. (X^s(u) \vee^s Y^s(u) \vee^s Z^s(u))) \wedge^s (\forall^s u. \forall^s v. (P^s(u,v) \supset^s ((\neg^s (X^s(u) \wedge^s X^s(v))) \wedge^s (\neg^s (Y^s(u) \wedge^s Y^s(v))) \wedge^s (\neg^s (Z^s(u) \wedge^s Z^s(v))))))))))$

unfolding *DefS apply (intro allI) apply simp nitpick oops*

lemma *vertex-cover-valid-s:*

$\models^{s'} (\exists^s_2 X. \forall^s u. \forall^s v. (P^s(u,v) \supset^s (X^s(u) \vee^s X^s(v))))$

unfolding *DefS*

apply *(intro allI, simp)*

by *(rule exI[of - Univ]) auto*

lemma *transitivity-not-valid-s:*

$\models^{s'} (\forall^s x. \forall^s y. \forall^s z. (P^s(x,y) \supset^s (P^s(y,z) \supset^s P^s(x,z))))$

unfolding *DefS apply (intro allI) apply simp nitpick oops*

lemma *symmetry-not-valid-s:*

$\models^{s'} (\forall^s x. \forall^s y. (P^s(x,y) \supset^s P^s(y,x)))$

unfolding *DefS apply (intro allI) apply simp nitpick oops*

lemma *triangle-exists-not-valid-s:*

$\models^{s'} (\exists^s x. \exists^s y. \exists^s z. (P^s(x,y) \wedge^s P^s(y,z) \wedge^s P^s(x,z)))$

unfolding *DefS apply (intro allI) apply simp nitpick oops*

lemma *has-edge-not-valid-s:*

$\models^{s'} (\exists^s x. \exists^s y. P^s(x,y))$

unfolding *DefS apply (intro allI) apply simp nitpick oops*

lemma *loop-free-not-valid-s:*

$\models^{s'} (\forall^s x. \neg^s P^s(x,x))$

unfolding *DefS apply (intro allI) apply simp nitpick oops*

9.6.3 Minimal-shallow embedding (under $\models^m$)

In the minimal embedding the second-order existential ranges over the (countable) assignment range, so even formulae that hold under the full second-order domain (such as vertex cover) need not hold here. *nitpick* can certify a POTENTIAL countermodel only.

lemma *three-colorability-not-valid-m:*

$\models^m (\exists^m_2 X. \exists^m_2 Y. \exists^m_2 Z. ((\forall^m u. (X^m(u) \vee^m Y^m(u) \vee^m Z^m(u))) \wedge^m (\forall^m u. \forall^m v. (P^m(u,v) \supset^m ((\neg^m (X^m(u) \wedge^m X^m(v))) \wedge^m (\neg^m (Y^m(u) \wedge^m Y^m(v))) \wedge^m (\neg^m (Z^m(u) \wedge^m Z^m(v))))))))))$

unfolding *DefM nitpick[expect=potential] oops*

lemma *vertex-cover-not-valid-m:*

$\models^m (\exists^m_2 X. \forall^m u. \forall^m v. (P^m(u,v) \supset^m (X^m(u) \vee^m X^m(v))))$

unfolding *DefM oops*

Explicit refutability of the vertex-cover schema in the minimal embedding: with II interpreting P as the complete relation and GG mapping every symbol to the empty set, no symbol-indexed X can cover any edge.

lemma *vertex-cover-refutable-m:*

$\exists (II'::\mathcal{I}) (gg'::\mathcal{E}) (GG'::\mathcal{G}).$
 $\neg (\exists X. \forall u. \forall v. II' P (gg' u) (gg' v) \longrightarrow (GG' X) (gg' u) \vee (GG' X) (gg' v))$
by (*rule* $exI[of - \lambda r a b. True]$,
rule $exI[of - \lambda -. undefined]$,
rule $exI[of - \lambda Z d. False]$) *simp*

lemma *transitivity-not-valid-m:*

$\models^m (\forall^m x. \forall^m y. \forall^m z. (P^m(x, y) \supset^m (P^m(y, z) \supset^m P^m(x, z))))$
unfolding *DefM nitpick oops*

lemma *symmetry-not-valid-m:*

$\models^m (\forall^m x. \forall^m y. (P^m(x, y) \supset^m P^m(y, x)))$
unfolding *DefM nitpick oops*

lemma *triangle-exists-not-valid-m:*

$\models^m (\exists^m x. \exists^m y. \exists^m z. (P^m(x, y) \wedge^m P^m(y, z) \wedge^m P^m(x, z)))$
unfolding *DefM oops*

Explicit refutability: with II interpreting P as the empty relation no triangle can exist.

lemma *triangle-exists-refutable-m:*

$\exists (II'::\mathcal{I}) (gg'::\mathcal{E}).$
 $\neg (\exists x. \exists y. \exists z. II' P (gg' x) (gg' y) \wedge II' P (gg' y) (gg' z) \wedge II' P (gg' x) (gg' z))$
by (*rule* $exI[of - \lambda r a b. False]$, *rule* $exI[of - \lambda -. undefined]$) *simp*

lemma *has-edge-not-valid-m:*

$\models^m (\exists^m x. \exists^m y. P^m(x, y))$
unfolding *DefM nitpick oops*

lemma *loop-free-not-valid-m:*

$\models^m (\forall^m x. \neg^m P^m(x, x))$
unfolding *DefM nitpick oops*

end

References

- [1] C. Ballarin. Locales: A module system for mathematical theories. *Journal of Automated Reasoning*, 52(2):123–153, 2014.
- [2] D. A. Basin and N. Klarlund. Hardware verification using monadic second-order logic. In P. Wolper, editor, *Computer Aided Verification*,

- 7th International Conference, CAV '95, Liège, Belgium, July 3–5, 1995, Proceedings*, volume 939 of *Lecture Notes in Computer Science*, pages 31–41. Springer, 1995.
- [3] C. Benzmüller, X. Parent, and L. van der Torre. Designing normative theories for ethical and legal reasoning: LogiKEy framework, methodology, and tool support. *Artificial Intelligence*, 287:103348, 2020.
 - [4] C. Benzmüller and L. C. Paulson. Multimodal and intuitionistic logics in simple type theory. *The Logic Journal of the IGPL*, 18(6):881–892, 2010.
 - [5] C. Benzmüller and L. C. Paulson. Quantified multimodal logics in simple type theory. *Logica Universalis (Special Issue on Multimodal Logics)*, 7(1):7–20, 2013.
 - [6] C. Benzmüller. Faithful logic embeddings in HOL — deep and shallow. In C. Barrett and U. Waldmann, editors, *Automated Deduction – CADE-30 – 30th International Conference on Automated Deduction, Stuttgart, Germany, July 28-31, 2025, Proceedings*, volume 15943 of *Lecture Notes in Computer Science*, pages 280–302. Springer, 2025. (preprint: arXiv:2502.19311).
 - [7] J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6(1–6):66–92, 1960.
 - [8] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*, volume 138 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2012.
 - [9] L. Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.
 - [10] M. Manzano. *Extensions of First-Order Logic*, volume 19 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1996.
 - [11] S. Shapiro. *Foundations without Foundationalism: A Case for Second-Order Logic*, volume 17 of *Oxford Logic Guides*. Oxford University Press, 1991.
 - [12] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. 3: Beyond Words*, pages 389–455. Springer, 1997.

- [13] D. Traytel and T. Nipkow. Decision procedures for MSO on words based on derivatives of regular expressions. *Archive of Formal Proofs*, June 2014. https://isa-afp.org/entries/MSO_Regex_Equivalence.html, Formal proof development.
- [14] J. Väänänen. *Models and Games*, volume 132 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 2011.