Elimination of Negation in a Logical Framework

Alberto Momigliano

June 30, 2000

Department of Philosophy
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:
Frank Pfenning, Chair
Dana Scott
Dale Miller, Pennsylvania State University

Abstract

We address the issue of endowing a logical framework with a logically justified notion of negation. Logical frameworks with a logic programming interpretation such as hereditary Harrop formulae cannot directly express negative information, although negation is a useful specification tool. Since negation-as-failure does not fit well in a logical framework, especially one endowed with hypothetical and parametric judgments, we adapt the idea of *elimination* of negation from Horn logic to a fragment of higher-order hereditary Harrop formulae. The idea is to replace occurrences of negative predicates with positive ones which are operationally equivalent. This entails two separate phases.

Complementing terms, i.e in our case higher-order patterns. Due the presence of *partially applied* lambda terms, intuitionistic lambda calculi are not closed under complementation. We thus develop a strict lambda calculus, where we can directly express whether a function depends on its argument.

Complementing clauses. This can be seen as a negation normal form procedure which is consistent with intuitionistic provability. It entails finding a middle ground between the Closed World Assumption usually associated with negation and the Open World Assumption typical of logical frameworks. As this is in general not possible, we restrict ourselves to a fragment in which clause complementation is viable and that has proven to be expressive enough for the practice of logical frameworks. The main technical idea is to isolate a set of programs where static and dynamic clauses do not overlap.

Contents

List of Figures

Chapter 1

Clause Complementation

In this chapter we introduce the source language and its uniform proofs system. Before formalizing the restriction to terminating programs, we establish the fundamental notion of context schema. This allows to enforce the *Regular World Assumption* (RWA), on which clause complementation is built. Finally, we discuss how to give an operational semantics to our system.

1.1 The Logic

We will use the following slightly unusual source language. Again, we fix a signature Σ in advance which would otherwise clutter the presentation; let $\vec{M}, \vec{N}$ be sequences of *simple* terms:

$$\begin{array}{rrcl} \textit{Atoms} & Q & ::= & q\ \vec{M} \mid \neg q\ \vec{M} \\ \textit{Goals} & G & ::= & Q \mid \top \mid \bot \mid \vec{M} \doteq \vec{N} \mid \vec{M} \neq \vec{N} \mid \\ & & & G_1 \wedge G_2 \mid G_1 \vee G_2 \mid D \rightarrow G \mid \forall x\!:\!A.\, G \mid \\ \textit{Clauses} & D & ::= & \top \mid \bot \mid Q \mid D \leftarrow G \mid D_1 \wedge D_2 \mid D_1 \vee D_2 \mid \forall x\!:\!A.\, D \end{array}$$

First and foremost, we restrict ourselves to clauses of at *most* third-order, that is we allow HHF which only make Horn assumptions. This simplifies the presentation of the complement algorithm, but it is not an unsurmountable obstacle; we comment on this and other restrictions in the Conclusions (Chapter ??). Differently form standard presentation of HHF, we do not allow existential goals. This rules out open queries and local variables, as well as spares us from the heavy machinery of *mixed prefixes* [?]. On the other hand, we allow disjunction between clauses, equality and inequality. While disjunction among clauses will be eliminated, inequalities will survive, although they will always be solvable at run-time by a simple syntactic check, as we will see in due time (Section ??). This simplifies their treatment in type-theoretic languages such as *Twelf*, where, in general, (in)equalities should be viewed as types inhabited by appropriate proof terms. The tokens $\top, \bot$ may be decorated with a predicate symbol, as explained in Subsection 6.1.2 and Definition 6.21. Finally, we remark that '$\neg$' is *not* a connective, but a name constructor for atomic formulae.

$$\begin{array}{rrcl} \textit{Parameter Contexts} & \Gamma & ::= & \cdot \mid \Gamma, x{:}A \\ \textit{Assumptions} & \mathcal{D} & ::= & \top \mid \mathcal{D} \wedge D \end{array}$$

We call a pair of concrete context and assumption $\Gamma; \mathcal{D}$, such that all parameters occurring in $\mathcal{D}$ are mentioned in Γ, a *(run-time) context.* We make the usual conventions on contexts, in particular we avoid mentioning the leading $\cdot$ and $\top$ elements.

We use the notation $D \sqsubseteq \mathcal{P} \wedge \mathcal{D}$ to indicate that D is a top-level conjunct in the $\mathcal{P}$ or in $\mathcal{D}$.1

We now introduce the uniform proofs judgments for provability and denial in Figure 6.1 and for immediate implication and denial in Figure 6.2. While the system for denial is introduced for technical reasons, namely the proof of the Exhaustivity Theorem ??, it is of independent interest, since it provide evidence (that is proof terms) for non-provability. A similar system was presented in [?], although in a simplified setting

$$\frac{}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}\top}\;\vdash\top \qquad \frac{}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}\bot}\;\nvdash\bot$$

$$\frac{\vec{M}=\vec{N}}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}\vec{M}\doteq\vec{N}}\;\vdash\doteq \qquad \frac{\vec{M}\neq\vec{N}}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}\vec{M}\doteq\vec{N}}\;\nvdash\doteq$$

$$\frac{\vec{M}\neq\vec{N}}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}\vec{M}\neq\vec{N}}\;\vdash\neq \qquad \frac{\vec{M}=\vec{N}}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}\vec{M}\neq\vec{N}}\;\nvdash\neq$$

$$\frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}G_1 \qquad \Gamma;\mathcal{D}\vdash_{\mathcal{P}}G_2}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}G_1\wedge G_2}\;\vdash\wedge \qquad \frac{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}G_1 \qquad \Gamma;\mathcal{D}\nvdash_{\mathcal{P}}G_2}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}G_1\vee G_2}\;\nvdash\vee$$

$$\frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}G_1}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}G_1\vee G_2}\;\vdash\vee_1 \qquad \frac{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}G_1}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}G_1\wedge G_2}\;\nvdash\wedge_1$$

$$\frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}G_2}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}G_1\vee G_2}\;\vdash\vee_2 \qquad \frac{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}G_2}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}G_1\wedge G_2}\;\nvdash\wedge_2$$

$$\frac{\Gamma;(\mathcal{D}\wedge D)\vdash_{\mathcal{P}}G}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}D\rightarrow G}\;\vdash\rightarrow \qquad \frac{\Gamma;(\mathcal{D}\wedge D)\nvdash_{\mathcal{P}}G}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}D\rightarrow G}\;\nvdash\rightarrow$$

$$\frac{(\Gamma,y{:}A);\mathcal{D}\vdash_{\mathcal{P}}[y/x]G}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}\forall x:A.\,G}\;\vdash\forall^{y} \qquad \frac{(\Gamma,y{:}A);\mathcal{D}\nvdash_{\mathcal{P}}[y/x]G}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}\forall x:A.\,G}\;\nvdash\forall^{y}$$

$$\frac{D\sqsubseteq\mathcal{P}\wedge\mathcal{D} \qquad \Gamma;\mathcal{D}\vdash_{\mathcal{P}}D\gg Q}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}}Q}\;\vdash\text{At} \qquad \frac{\text{for every }D\sqsubseteq\mathcal{P}\wedge\mathcal{D} \qquad \Gamma;\mathcal{D}\nvdash_{\mathcal{P}}D\gg Q}{\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}Q}\;\nvdash\text{At}$$

Figure 1.1: Provability and denial

without parametric judgments. We remark that the systems for denial will instead not be needed in the proof of exclusivity (Theorem 6.34).

Due to the presence of '$\vee$' as a clause constructor, uniform proofs are *not* complete w.r.t. intuitionistic logic. We will remedy this situation in Section ??. The rules are depicted (when possible) in two columns where every row displays a positive rule and its negative counterpart.

$\Gamma;\mathcal{D}\vdash_{\mathcal{P}}G$	Program $\mathcal{P}$ and assumption $\mathcal{D}$ uniformly entail G.
$\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}G$	Program $\mathcal{P}$ and assumption $\mathcal{D}$ uniformly deny G.
$\Gamma;\mathcal{D}\vdash_{\mathcal{P}}D\gg Q$	Clause D from $\mathcal{P}\wedge\mathcal{D}$ immediately entails atom Q.
$\Gamma;\mathcal{D}\nvdash_{\mathcal{P}}D\gg Q$	Clause D from $\mathcal{P}\wedge\mathcal{D}$ immediately denies atom Q.

We briefly comment on the rules: the (in)equalities rules simply mirror the object logic symbols $\doteq,\neq$ in meta-level (in)equality. The denial rules for implication and universal quantification reflect the operational semantics of unprovability that we have discussed earlier. Note that $\not\gg\forall$ is an infinitary rule, due to the meta-linguistic extensional universal quantification on all terms. Rule $\not\gg\forall$ must be read as: for every well-typed term n in parameter context Γ, $\forall x:A.\,D$ immediately denies Q if so does $[n/x]D$. Rules $\vdash\forall$, $\nvdash\forall$ are instead parametric in the new eigenvariable, say y; the $()^{y}$ superscript reminds us of the parameter condition on y, e.g. that y does not occur free in $\Gamma;\mathcal{D}$ nor in G. We will use this notation in any other parametric rule.

Differently from the latter, we will also use *global* eigenvariables, say u (see for example rule $\stackrel{D}{\Longrightarrow}\forall$ in Figure 6.3); this expresses the fact that in those rules the relation holds for *any* term; those parameters are therefore analogous to logic variables. We will pervasively utilize this notation; however, to avoid notational

$$\frac{}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bot \gg Q} \gg\bot \qquad \frac{}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \top \gg Q} \not\gg\top$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [t/x]D \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall x : A.\, D \gg Q} \gg\forall$$

$$\frac{\text{for all } n \ \Gamma \vdash n : A \qquad \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [n/x]D \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \forall x : A.\, D \gg Q} \not\gg\forall$$

$$\frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \wedge D_2 \gg Q} \gg\wedge_1 \qquad \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \vee D_2 \gg Q} \not\gg\vee_1$$

$$\frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \wedge D_2 \gg Q} \gg\wedge_2 \qquad \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \vee D_2 \gg Q} \not\gg\vee_2$$

$$\frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \vee D_2 \gg Q} \gg\vee$$

$$\frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q \qquad \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \wedge D_2 \gg Q} \not\gg\wedge$$

$$\frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} G}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \leftarrow G \gg Q} \gg\rightarrow$$

$$\frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \leftarrow G \gg Q} \not\gg\rightarrow_1 \qquad \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \leftarrow G \gg Q} \not\gg\rightarrow_2$$

Figure 1.2: Immediate entailment and denial

clutter, when possible we shall not keep an explicit record of global parameters, but we agree to implicitly gather them in a pool Φ, which we can access when needed, in particular to type-check substitutions, see next 6.1.1.

For the sake of conciseness we will often use in this thesis the following rule schema, where $\mathcal{J}(X, R)$ is a judgment involving a token X ranging over a set $\mathcal{T}$ and a relation R:

$$\frac{X \in \mathcal{T}}{\mathcal{J}(X, R)}\ R\ X$$

1.1.1 Substitutions

We will need the notion of *well-typed substitutions*:

$$\textit{Substitutions} \quad \sigma \quad ::= \epsilon \mid \sigma, t/x$$

For $\sigma = \sigma', t/x$, we say that x is *defined* in σ. We require all variables defined in a substitution to be distinct: we use $\mathrm{dom}(\sigma)$ for the set of variables defined in σ and $\mathrm{cod}(\sigma)$ for the variables occurring in the substituting terms. We assume them to be disjoint. Notation: $[t/x]E$ denotes the substitution of x with t in an well-formed expression E. More in general we denote with $[\sigma]E$ the application of σ to E. We will also need an operation of composition, say $\sigma_1 \cdot \sigma_2$, such that $[\sigma_1 \cdot \sigma_2]E = [\sigma_2]([\sigma_1]E)$.

The judgment $\Gamma \vdash \sigma : \Phi$ formalizes that σ is a valid well-typed substitution w.r.t. Γ.

$$\frac{}{\cdot \vdash \epsilon : \cdot}\ \epsilon\cdot \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash \sigma : \Phi}{\Gamma \vdash \sigma, t/x : (\Phi, x : A)}\ \sigma\Phi$$

1.1.2 $\top$-Normalization

We show in this section how to put every program in a normalized format w.r.t. assumptions so that every goal in the scope of an universal quantifier depends on some assumption, possibly the trivial clause $\top$. This has also the effect of 'localizing' the trivial assumption to its atom, a property will be very useful while performing augmentation, see Section 6.6.

We first state some basic properties of provability.

Lemma 1.1 (Weakening)

1. *If* $\Gamma; \mathcal{D} \vdash G$, *then:*
   (a) $(\Gamma, x{:}A); \mathcal{D} \vdash G$.
   (b) $\Gamma; (\mathcal{D} \wedge D) \vdash G$.
2. *If* $\Gamma; \mathcal{D} \vdash D \gg Q$, *then:*
   (a) $(\Gamma, x{:}A); \mathcal{D} \vdash D \gg Q$.
   (b) $\Gamma; (\mathcal{D} \wedge D') \vdash D \gg Q$.

Proof: A straightforward mutual induction on the given derivations. □

In the proof of Theorem 6.5 we will need the following form of Strengthening:

Lemma 1.2 ($\top$-Strengthening)

1. *If* $\Gamma; (\mathcal{D} \wedge \top) \vdash G$, *then* $\Gamma; \mathcal{D} \vdash G$.
2. *If* $\Gamma; (\mathcal{D} \wedge \top) \vdash D \gg Q$, *then* $\Gamma; \mathcal{D} \vdash D \gg Q$.

Proof: By an easy mutual induction on the structure of the given derivations. □

Normalization is realized in Figure 6.3 by the two judgments $D \stackrel{D^\top}{\Longrightarrow} D^\top$ and $\Gamma; \mathcal{D} \vdash G \stackrel{G^\top}{\Longrightarrow} G^\top$. The only interesting case is when we reach an atomic goal: if no parameter has been introduced, the clause does not require any normalization. If $\Gamma; \mathcal{D}$ is non-empty, we adopt a brute-force approach and add the trivial clause for *every* predicate in the signature. This is largely unnecessary, but gives a very simple account of mutually recursion, without an explicit appeal to a call graph. As a matter of fact, we would really need to $\top$-normalize an atom Q with the trivial clause for every predicate that is mutually recursive to Q and such the type of the parameter in the context is not *subordinate* [?] to the type of Q. On the other hand Theorems 6.4 and 6.5 ensures, as obvious, that this program transformation is harmless, since it preserves provability and denial. Moreover, we will engineer the rule for clause complementation so that the irrelevant $\top_q$ will be inactive. We will remove those irrelevant augmentations in a final pass (Section ??).

Example 1.3 *Consider the lambda clause in the* `open` *program:*

$$\begin{aligned} oplam \quad &: \quad \forall E\!:\!exp \rightarrow exp.\, open\ (lam\ E) \\ &\qquad \leftarrow \forall x\!:\!exp.\, open\ (E\ x). \end{aligned}$$

$oplam \stackrel{D^\top}{\Longrightarrow} oplam^\top$, i.e.

$$\begin{aligned} oplam^\top \quad &: \quad \forall E\!:\!exp \rightarrow exp.\, open\ (lam\ E) \\ &\qquad \leftarrow (\forall x\!:\!exp.\, \top_{open} \rightarrow open\ (E\ x)). \end{aligned}$$

Consider this fragment of code involving two of mutually recursive predicates:

$$\begin{aligned} qlam \quad &: \quad \forall E\!:\!exp \rightarrow exp.\, q\ (lam\ E) \\ &\qquad \leftarrow \forall x\!:\!exp.\, p\ (E\ x). \\ papp \quad &: \quad \forall E_1\!:\!exp.\forall E_2\!:\!exp.\, p\ (app\ E_1\ E_2) \\ &\qquad \leftarrow q\ E_1 \wedge p\ E_2. \end{aligned}$$

Then:

$$\begin{aligned} qlam^\top \quad &: \quad \forall E\!:\!exp \rightarrow exp.\, q\ (lam\ E) \\ &\qquad \leftarrow (\forall x\!:\!exp.\, \top_p \wedge \top_q \rightarrow p\ (E\ x)). \end{aligned}$$

In examples, we will not mention inactive $\top$ clauses.

We start by showing that this transformation preserves run-time provability.

Theorem 1.4 *Let $\Gamma'; \mathcal{D}' \vdash G \stackrel{G^\top}{\Longrightarrow} G^\top$, $D \stackrel{D^\top}{\Longrightarrow} D^\top$, $\mathcal{D} \stackrel{D^\top}{\Longrightarrow} \mathcal{D}^\top$ and $\Gamma \vdash \theta : \Phi$.*

1. *If $\Gamma; [\theta]\mathcal{D} \vdash [\theta]G$, then $\Gamma; [\theta]\mathcal{D}^\top \vdash [\theta]G^\top$.*
2. *If $\Gamma; [\theta]\mathcal{D} \vdash [\theta]D \gg [\theta]Q$, then $\Gamma; [\theta]\mathcal{D}^\top \vdash [\theta]D^\top \gg [\theta]Q$.*

Proof: By mutual induction on the structure of the derivation of $\gamma :: \Gamma; [\theta]\mathcal{D} \vdash [\theta]G$ and inversion on $\gamma' :: \Gamma'; \mathcal{D}' \vdash G \stackrel{G^\top}{\Longrightarrow} G^\top$ together with $\delta :: \Gamma; [\theta]\mathcal{D} \vdash [\theta]D \gg [\theta]Q$ and inversion on $\delta' :: D \stackrel{D^\top}{\Longrightarrow} D^\top$. We sketch only some cases:

Case: γ ends in $\vdash$ At and γ' is $\stackrel{G^\top}{\Longrightarrow} X, \stackrel{G^\top}{\Longrightarrow}$ At-emp: trivial.

Case: γ ends in $\vdash$ At and γ' ends in $\stackrel{G^\top}{\Longrightarrow}$ At:

$\Gamma; [\theta]\mathcal{D} \vdash [\theta]Q$	By hypothesis
$\Gamma; [\theta]\mathcal{D} \vdash [\theta]D \gg [\theta]Q$	By sub-derivation
$\Gamma; [\theta]\mathcal{D}^\top \vdash [\theta]D^\top \gg [\theta]Q$	By IH 2
$\Gamma; ([\theta]\mathcal{D}^\top \wedge (\bigwedge_{p\in\Sigma} \top_p)) \vdash [\theta]D^\top \gg [\theta]Q$	By repeated Weakening
$\Gamma; ([\theta]\mathcal{D}^\top \wedge (\bigwedge_{p\in\Sigma} \top_p)) \vdash [\theta]Q$	By rule $\vdash$ At
$\Gamma; [\theta]\mathcal{D}^\top \vdash (\bigwedge_{p\in\Sigma} \top_p) \rightarrow [\theta]Q$	By rule $\vdash\rightarrow$
$\Gamma; [\theta]\mathcal{D} \vdash [\theta]Q^\top$	By rule $\stackrel{G^\top}{\Longrightarrow}$ At

$$\frac{X \in \{\top, \bot\} \text{ or (dis)eq}}{\Gamma; \mathcal{D} \vdash X \overset{G^\top}{\Longrightarrow} X} \overset{G^\top}{\Longrightarrow} X$$

$$\frac{}{\cdot; \top \vdash Q \overset{G^\top}{\Longrightarrow} Q} \overset{G^\top}{\Longrightarrow} \text{At-emp} \qquad \frac{}{\Gamma; \mathcal{D} \vdash Q \overset{G^\top}{\Longrightarrow} (\bigwedge_{p \in \Sigma} \top_p \to Q)} \overset{G^\top}{\Longrightarrow} \text{At}$$

$$\frac{\Gamma; (\mathcal{D} \wedge D) \vdash G \overset{G^\top}{\Longrightarrow} G^\top \qquad D \overset{D^\top}{\Longrightarrow} D^\top}{\Gamma; \mathcal{D} \vdash D \to G \overset{G^\top}{\Longrightarrow} D^\top \to G^\top} \overset{G^\top}{\Longrightarrow}\to$$

$$\frac{(\Gamma, y{:}A); \mathcal{D} \vdash [y/x]G \overset{G^\top}{\Longrightarrow} [y/x]G^\top}{\Gamma; \mathcal{D} \vdash \forall x : A.\, G \overset{G^\top}{\Longrightarrow} \forall x : A.\, G^\top} \overset{G^\top}{\Longrightarrow} \forall^y$$

$$\frac{\Gamma; \mathcal{D} \vdash G_1 \overset{G^\top}{\Longrightarrow} G_1^\top \qquad \Gamma; \mathcal{D} \vdash G_2 \overset{G^\top}{\Longrightarrow} G_2^\top}{\Gamma; \mathcal{D} \vdash G_1 \wedge G_2 \overset{G^\top}{\Longrightarrow} G_1^\top \wedge G_2^\top} \overset{G^\top}{\Longrightarrow} \wedge$$

$$\frac{\Gamma; \mathcal{D} \vdash G_1 \overset{G^\top}{\Longrightarrow} G_1^\top \qquad \Gamma; \mathcal{D} \vdash G_2 \overset{G^\top}{\Longrightarrow} G_2^\top}{\Gamma; \mathcal{D} \vdash G_1 \vee G_2 \overset{G^\top}{\Longrightarrow} G_1^\top \vee G_2^\top} \overset{G^\top}{\Longrightarrow} \vee$$

$$\frac{X \in \{Q, \top, \bot\}}{X \overset{D^\top}{\Longrightarrow} X} \overset{D^\top}{\Longrightarrow} X$$

$$\frac{\cdot; \top \vdash G \overset{G^\top}{\Longrightarrow} G^\top \qquad D \overset{D^\top}{\Longrightarrow} D^\top}{(G \to D) \overset{D^\top}{\Longrightarrow} G^\top \to D^\top} \overset{D^\top}{\Longrightarrow}\to$$

$$\frac{[u/x]D \overset{D^\top}{\Longrightarrow} [u/x]D^\top}{\forall x : A.\, D \overset{D^\top}{\Longrightarrow} \forall x : A.\, D^\top} \overset{D^\top}{\Longrightarrow} \forall^u$$

$$\frac{D_1 \overset{D^\top}{\Longrightarrow} D_1^\top \qquad D_2 \overset{D^\top}{\Longrightarrow} D_2^\top}{D_1 \wedge D_2 \overset{D^\top}{\Longrightarrow} D_1^\top \wedge D_2^\top} \overset{D^\top}{\Longrightarrow} \wedge \qquad \frac{D_1 \overset{D^\top}{\Longrightarrow} D_1^\top \qquad D_2 \overset{D^\top}{\Longrightarrow} D_2^\top}{D_1 \vee D_2 \overset{D^\top}{\Longrightarrow} D_1^\top \vee D_2^\top} \overset{D^\top}{\Longrightarrow} \vee$$

Figure 1.3: $\top$-Normalization

The other cases follows immediately by IH 1 and 2.

□

Now, the converse:

Theorem 1.5 *Let* $\Gamma'; \mathcal{D}' \vdash G \overset{G^\top}{\Longrightarrow} G^\top$, $D \overset{D^\top}{\Longrightarrow} D^\top$, $\mathcal{D} \overset{D^\top}{\Longrightarrow} \mathcal{D}^\top$ *and* $\Gamma \vdash \theta : \Phi$.

1. *If* $\Gamma; [\theta]\mathcal{D}^\top \vdash [\theta]G^\top$, *then* $\Gamma; [\theta]\mathcal{D} \vdash [\theta]G$.
2. *If* $\Gamma; [\theta]\mathcal{D}^\top \vdash [\theta]D^\top \gg [\theta]Q$, *then* $\Gamma; [\theta]\mathcal{D} \vdash [\theta]D \gg [\theta]Q$.

Proof: By mutual induction on the structure of $\gamma :: \Gamma; [\theta]\mathcal{D}^\top \vdash [\theta]G^\top$ and $\delta :: \Gamma; [\theta]\mathcal{D}^\top \vdash [\theta]D^\top \gg [\theta]Q$, as above but using $\top$-Strengthening (Lemma 6.2) in place of Weakening. □

An analogous result can be proven w.r.t. failure; the latter would require $\top$-Weakening and Strengthening w.r.t. the (immediate) denial relation. On the other hand, the above result will suffice, if we restrict to terminating programs as we will do in Section 6.3. First, we need to establish a theory of *context schemata*

1.2 Context Schemata

In this Section we address the properties of contexts[1].

Hereditary Harrop formulae differ from Horn clauses in that they may dynamically extend the current signature and program; this is reflected in the provability (and denial) relation which is parameterized by a run-time context $\Gamma; \mathcal{D}$. As we have argued before (Section 5.5) we cannot obtain closure under clause complementation for the full logic of HHF, but we have to restrict ourselves to a smaller (but significant) fragment. This in turn entails that we have to make sure that during execution whenever an assumption is made, the latter stays in the fragment we have isolated. Technically, we proceed as follows:

- We extract from the static definition of a predicate the general 'template' of a legal assumption.
- We require dynamic assumptions to conform to this template.

We thus introduce the notion of *schema satisfaction*, for which we will need the following data structure: a *schema context* abstracts over all possible instantiation of a run-time context. To account for that, we introduce a quantifier-like operator, say SOME $\Phi.\mathcal{D}$, which takes a clause and existentially bounds its free variables, i.e. the global parameters occurring in $\mathcal{D}$. We use the following reification of schemata: do not confuse '$\|$', which is used to represent schema alternatives in the object language with '$|$' which does the same in our informal meta-language.

$$\textit{Contexts Schemata} \quad \mathcal{S} \quad ::= \quad \circ \mid \mathcal{S} \| (\Gamma; \text{SOME } \Phi.\mathcal{D})$$

Example 1.6 *The schema context grammar induced by the* `linear` *predicate (Example 5.2) is as follows:*

$$\mathcal{S}_{linear} \quad = \quad \circ \mid \mathcal{S}_{linear} \| x{:}exp; linear(x) \mid \mathcal{S}_{linear} \| x{:}exp; \top_{linx}$$

while a possible run-time context is

$$y_1{:}exp, y_2{:}exp, y_3{:}exp; \top_{linx} \wedge \top_{linx} \wedge linear(y_3)$$

Since every possible assumption is closed, there is no existential binding. Consider instead the clauses for type-checking in the simply typed calculus:

$$\begin{array}{lll} ofapp & : & \forall E_1, E_2 : exp. \forall T_1, T_2, T : tp. \\ & & \quad of\ (app\ E_1\ E_2)\ T_2 \\ & & \qquad \leftarrow of\ E_1\ (arrow\ T_1\ T_2) \\ & & \qquad \leftarrow of\ E_2\ T_1. \\ oflam & : & \forall E : exp \rightarrow exp. \forall T_1, T_2 : tp. \\ & & \quad of\ lam(E)\ (arrow\ T_1\ T_2) \\ & & \qquad \leftarrow (\forall x : exp.\ of\ x\ T_1 \rightarrow of\ (E\ x)\ T_2). \end{array}$$

[1] This section is inspired by Schürmann's treatment of similar material in his dissertation [?].

Then the schema context is:

$$\mathcal{S}_{of} \quad = \quad \circ \mid \mathcal{S}_{of} \| x{:}exp; SOME\ u : tp.\ of\ x\ u$$

where 'u' is a global parameter.

We will also need to disambiguate blocks in run-time contexts; overlapping may indeed happen when the alternatives in a context schema are not disjoint. For example, suppose we are given the following schema and partial run-time context:

$$\mathcal{S} \quad = \quad \circ \mid \mathcal{S} \| x{:}exp; p(x) \mid \mathcal{S} \| x{:}exp; p(x) \wedge q(x)$$

$$y_1{:}exp, y_2{:}exp; p(y_1) \wedge p(y_2)$$

It is not uniquely determined whether, say, $p(y_2)$ is an instance of the first alternative or a prefix of the second one. To determine that, we simply modify the provability and denial rules $\vdash$ At and $\nvdash$ At by 'packaging' run-time contexts into blocks with a bracket operator $\lceil \_ \rceil$. Intuitively, a block is complete when an atomic conclusion is reached during the deduction. It is therefore bracketed when the run-time context is passed to the immediate implication judgment.

$$\frac{D \sqsubseteq \mathcal{P} \wedge \mathcal{D} \qquad \lceil \Gamma \rceil ; \lceil \mathcal{D} \rceil \vdash_{\mathcal{P}} D \gg Q}{\Gamma ; \mathcal{D} \vdash_{\mathcal{P}} Q} \vdash \mathrm{At} \qquad \frac{\text{for every } D \sqsubseteq \mathcal{P} \wedge \mathcal{D} \qquad \lceil \Gamma \rceil ; \lceil \mathcal{D} \rceil \nvdash_{\mathcal{P}} D \gg Q}{\Gamma ; \mathcal{D} \nvdash_{\mathcal{P}} Q} \nvdash \mathrm{At}$$

Still, we need to 'flatten' blocks, so that re-bracketing during execution will be immaterial on already delimited blocks; we thus require the bracket operator to satisfy the following absorption-distribution law:

$$\begin{aligned} \lceil \lceil \Gamma \rceil , \Gamma' \rceil &= \lceil \Gamma \rceil , \lceil \Gamma' \rceil \\ \lceil \lceil \mathcal{D} \rceil \wedge \mathcal{D}' \rceil &= \lceil \mathcal{D} \rceil \wedge \lceil \mathcal{D}' \rceil \end{aligned}$$

We are now ready to introduce schema satisfaction; this is accomplished by the following judgments:

- $\models_{\mathcal{S}} D$: clause D satisfies schema $\mathcal{S}$.
- $\Gamma ; \mathcal{D} \setminus G < \mathcal{S}$: (partially completed) run-time context $\Gamma ; \mathcal{D}$ together with goal G satisfies schema $\mathcal{S}$.
- $\Gamma ; \mathcal{D} < \mathcal{S}$: (completed) run-time context $\Gamma ; \mathcal{D}$ satisfies schema $\mathcal{S}$.
- $\Gamma \vdash \Gamma' ; \mathcal{D}' \in \mathcal{S}$: block $\Gamma' ; \mathcal{D}'$ in context Γ occurs in schema $\mathcal{S}$.

First, we say that a completed block *occurs* in a schema when the block is an alphabetic variant of some instantiation of one of the alternatives of the schema;

$$\frac{\Gamma \vdash \theta : \Phi \qquad (\Gamma' ; \mathcal{D}') \equiv_{\alpha} (\Gamma'' ; [\theta]\mathcal{D})}{\Gamma \vdash (\Gamma' ; \mathcal{D}') \in \mathcal{S} \| (\Gamma'' ; \mathrm{SOME}\ \Phi . \mathcal{D})} \in_1 \qquad \frac{\Gamma \vdash (\Gamma' ; \mathcal{D}') \in \mathcal{S}}{\Gamma \vdash (\Gamma' ; \mathcal{D}') \in \mathcal{S} \| (\Gamma'' ; \mathcal{D})} \in_2$$

Let us now analyze the rules in Figure 6.4. The empty run-time context is an instance of every schema. Moreover if Γ' and $\mathcal{D}'$ are completed blocks which occur in $\mathcal{S}$, as denoted by the block notation, then $(\Gamma, \lceil \Gamma' \rceil) ; (\mathcal{D} \wedge \lceil \mathcal{D}' \rceil)$ is an instance of $\mathcal{S}$, provided that $\mathcal{D}'$ is a valid clause. Thus we need to define the instance relation simultaneously to clause (and in turn to goal) satisfaction. The judgment $\models_{\mathcal{S}} D$ is merely auxiliary to the $\Gamma ; \mathcal{D} \setminus G < \mathcal{S}$ one: here we mimic the construction on the run-time schema until, in the base case, we check whether the resulting context is an instance of the given schema.

The following Lemma ensures that when an assumption D is pulled from a legal run-time context, i.e. which satisfies a given schema, so does D.

Lemma 1.7 *If* $\Gamma ; \mathcal{D} < \mathcal{S}$ *and* $D \sqsubseteq \mathcal{D}$*, then* $\models_{\mathcal{S}} D$.

Proof: By induction on the structure of the derivation of $\pi :: \Gamma ; \mathcal{D} < \mathcal{S}$.

$$\frac{X \in \{Q, \top, \bot\} \text{ or (dis)eq} \qquad \Gamma; \mathcal{D} < \mathcal{S}}{\Gamma; \mathcal{D} \setminus X < \mathcal{S}} \setminus X$$

$$\frac{\Gamma; \mathcal{D} \setminus G_1 < \mathcal{S} \qquad \Gamma; \mathcal{D} \setminus G_2 < \mathcal{S}}{\Gamma; \mathcal{D} \setminus G_1 \wedge G_2 < \mathcal{S}} \setminus\wedge$$

$$\frac{\Gamma; \mathcal{D} \setminus G_1 < \mathcal{S} \qquad \Gamma; \mathcal{D} \setminus G_2 < \mathcal{S}}{\Gamma; \mathcal{D} \setminus G_1 \vee G_2 < \mathcal{S}} \setminus\vee$$

$$\frac{(\Gamma, y{:}A); \mathcal{D} \setminus [y/x]G < \mathcal{S}}{\Gamma; \mathcal{D} \setminus \forall x{:}A.\, G < \mathcal{S}} \setminus\forall^y$$

$$\frac{\models_{\mathcal{S}} D \qquad \Gamma; (\mathcal{D} \wedge D) \setminus G < \mathcal{S}}{\Gamma; \mathcal{D} \setminus D \to G < \mathcal{S}} \setminus \to$$

$$\frac{X \in \{Q, \top\bot\}}{\models_{\mathcal{S}} X} \models_{\mathcal{S}} X$$

$$\frac{\models_{\mathcal{S}} D_1 \qquad \models_{\mathcal{S}} D_2}{\models_{\mathcal{S}} D_1 \wedge D_2} \models_{\mathcal{S}} \wedge$$

$$\frac{\models_{\mathcal{S}} D_1 \qquad \models_{\mathcal{S}} D_2}{\models_{\mathcal{S}} D_1 \vee D_2} \models_{\mathcal{S}} \vee$$

$$\frac{\models_{\mathcal{S}} [u/x]D}{\models_{\mathcal{S}} \forall x{:}A.\, D} \models_{\mathcal{S}} \forall^u$$

$$\frac{\models_{\mathcal{S}} D \qquad \cdot; \top \setminus G < \mathcal{S}}{\models_{\mathcal{S}} G \to D} \models_{\mathcal{S}}\to$$

$$\frac{}{\cdot; \top < \mathcal{S}} <_1 \qquad \frac{\Gamma \vdash (\Gamma'; \mathcal{D}') \in \mathcal{S} \quad \models_{\mathcal{S}} \mathcal{D}' \quad (\Gamma; \mathcal{D}) < \mathcal{S}}{(\Gamma, \lceil \Gamma' \rceil); (\mathcal{D} \wedge \lceil \mathcal{D}' \rceil) < \mathcal{S}} <_2$$

Figure 1.4: Judgments $\Gamma; \mathcal{D} \setminus G < \mathcal{S}$, $\models_{\mathcal{S}} D$ and $\Gamma; \mathcal{D} < \mathcal{S}$

Case:

$$\pi = \frac{}{\cdot;\top < \mathcal{S}} <_1$$

Then $D = \top$ and immediately

$$\pi' = \frac{}{\cdot;\top \models_{\mathcal{S}} \top} \models_{\mathcal{S}} \top$$

Case:

$$\pi = \frac{\Gamma'' \vdash (\Gamma';\mathcal{D}') \in \mathcal{S} \quad \models_{\mathcal{S}} \mathcal{D}' \quad (\Gamma'';\mathcal{D}'') < \mathcal{S}}{(\Gamma'', \lceil \Gamma' \rceil);(\mathcal{D}'' \wedge \lceil \mathcal{D}' \rceil) < \mathcal{S}} <_2$$

Subcase: $D \equiv \mathcal{D}'$
$\models_{\mathcal{S}} \mathcal{D}'$ By hypothesis

Subcase: $D \sqsubseteq \mathcal{D}''$
$\Gamma'';\mathcal{D}'' < \mathcal{S}$ By sub-derivation
$\models_{\mathcal{S}} D$ By IH

□

We can also show that schema satisfaction is closed under substitution:

Lemma 1.8 *Let* $\Gamma;\mathcal{D} < \mathcal{S}$ *and* $\Gamma \vdash \theta : \Phi$.

1. *If* $\models_{\mathcal{S}} D$, *then* $\models_{\mathcal{S}} [\theta]D$.
2. *If* $\Gamma;\mathcal{D} \setminus G < \mathcal{S}$, *then* $\Gamma;[\theta]\mathcal{D} \setminus [\theta]G < \mathcal{S}$.
3. *If* $\Gamma;\mathcal{D} < \mathcal{S}$ *then* $\Gamma;[\theta]\mathcal{D} < \mathcal{S}$.

Proof: By a straightforward mutual induction on the structure of $\pi :: \models_{\mathcal{S}} D$, $\gamma :: \Gamma;\mathcal{D} \setminus G < \mathcal{S}$ and $\sigma :: \Gamma;\mathcal{D} < \mathcal{S}$. □

1.2.1 Schema Extraction

Roughly, we extract a context schema by collecting all negative occurrences in a goal. In fact, those will be dynamically added to the static program under evaluation. This is achieved by the judgment $\Gamma;\mathcal{D} \vdash_{\Phi} G \stackrel{G}{\Longrightarrow} \mathcal{S}$. The latter is mutually recursive to the judgment $\vdash_{\Phi} D \stackrel{D}{\Longrightarrow} \mathcal{S}$ which collects schemata for each clause in the given program, as depicted in Figure 6.5. When an atomic goal is reached, the current list of parameters and assumptions is returned, with the correct existential binding inferred from the context of global variables. With the notation $\Phi_{\mathcal{D}}$ we mean the restriction of context Φ to the free variables of $\mathcal{D}$, i.e. $\{u{:}A \mid u \in \mathrm{dom}(\Phi), u \notin FV(\mathcal{D})\}$. We make the convention to absorb schema alternatives which are α-variants. Moreover the empty run-time context behaves as a left and right zero element for $||$, i.e. :

$$(\Gamma;\mathcal{D}||\cdot;\top) = (\cdot;\top||\Gamma;\mathcal{D}) = \Gamma;\mathcal{D}$$

Finally $||$ is commutative.

Example 1.9 *If* $\mathcal{P}$ *is the program in Example ??,* $\mathcal{P} \stackrel{D}{\Longrightarrow} x{:}exp; linear(x)||x{:}exp; \top_{linx}$.

We aim to show that if a schema context is extracted by a program, the latter satisfies the former. First, we need to establish two weakening lemmata w.r.t. schema alternatives:

Lemma 1.10 *If* $\Gamma \vdash \Gamma';\mathcal{D}' \in \mathcal{S}$ *then* $\Gamma \vdash \Gamma';\mathcal{D}' \in S'||S$.

Proof: By a straightforward induction on the structure of $\pi :: \Gamma;\mathcal{D} \in \mathcal{S}$. □

$$\frac{X \in \{Q, \top, \bot\} \text{ or (dis)eq}}{\Gamma; \mathcal{D} \vdash_\Phi X \stackrel{G}{\Longrightarrow} \Gamma; \text{SOME } \Phi_{\mathcal{D}}.\, \mathcal{D}} \stackrel{G}{\Longrightarrow} X$$

$$\frac{\Gamma; (\mathcal{D} \wedge D) \vdash_\Phi G \stackrel{G}{\Longrightarrow} \mathcal{S}_1 \qquad D \stackrel{D}{\Longrightarrow} \mathcal{S}_2}{\Gamma; \mathcal{D} \vdash_\Phi D \rightarrow G \stackrel{G}{\Longrightarrow} \mathcal{S}_1 || \mathcal{S}_2} \stackrel{G}{\Longrightarrow}\rightarrow$$

$$\frac{(\Gamma, y{:}A); \mathcal{D} \vdash_\Phi [y/x]G \stackrel{G}{\Longrightarrow} \mathcal{S}}{\Gamma; \mathcal{D} \vdash_\Phi \forall x : A.\, G \stackrel{G}{\Longrightarrow} \mathcal{S}} \stackrel{G}{\Longrightarrow} \forall^y$$

$$\frac{\Gamma; \mathcal{D} \vdash_\Phi G_1 \stackrel{G}{\Longrightarrow} \mathcal{S}_1 \qquad \Gamma; \mathcal{D} \vdash_\Phi G_2 \stackrel{G}{\Longrightarrow} \mathcal{S}_2}{\Gamma; \mathcal{D} \vdash_\Phi G_1 \wedge G_2 \stackrel{G}{\Longrightarrow} \mathcal{S}_1 || \mathcal{S}_2} \stackrel{G}{\Longrightarrow} \wedge$$

$$\frac{\Gamma; \mathcal{D} \vdash_\Phi G_1 \stackrel{G}{\Longrightarrow} \mathcal{S}_1 \qquad \Gamma; \mathcal{D} \vdash_\Phi G_2 \stackrel{G}{\Longrightarrow} \mathcal{S}_2}{\Gamma; \mathcal{D} \vdash_\Phi G_1 \vee G_2 \stackrel{G}{\Longrightarrow} \mathcal{S}_1 || \mathcal{S}_2} \stackrel{G}{\Longrightarrow} \vee$$

$$\frac{X \in \{Q, \top, \bot\}}{\vdash_\Phi X \stackrel{D}{\Longrightarrow} \cdot; \top} \stackrel{D}{\Longrightarrow} X$$

$$\frac{\cdot; \top \vdash_\Phi G \stackrel{G}{\Longrightarrow} \mathcal{S}_1 \qquad \vdash_\Phi D \stackrel{D}{\Longrightarrow} \mathcal{S}_2}{\vdash_\Phi (G \rightarrow D) \stackrel{D}{\Longrightarrow} \mathcal{S}_1 || \mathcal{S}_2} \stackrel{D}{\Longrightarrow}\rightarrow$$

$$\frac{\vdash_{\Phi, u:A} [u/x]D \stackrel{D}{\Longrightarrow} \mathcal{S}}{\vdash_\Phi \forall x : A.\, D \stackrel{D}{\Longrightarrow} \mathcal{S}} \stackrel{D}{\Longrightarrow} \forall^u$$

$$\frac{\vdash_\Phi D_1 \stackrel{D}{\Longrightarrow} \mathcal{S}_1 \qquad \vdash_\Phi D_2 \stackrel{D}{\Longrightarrow} \mathcal{S}_2}{\vdash_\Phi D_1 \wedge D_2 \stackrel{D}{\Longrightarrow} \mathcal{S}_1 || \mathcal{S}_2} \stackrel{D}{\Longrightarrow} \wedge$$

$$\frac{\vdash_\Phi D_1 \stackrel{D}{\Longrightarrow} \mathcal{S}_1 \qquad \vdash_\Phi D_2 \stackrel{D}{\Longrightarrow} \mathcal{S}_2}{\vdash_\Phi D_1 \vee D_2 \stackrel{D}{\Longrightarrow} \mathcal{S}_1 || \mathcal{S}_2} \stackrel{D}{\Longrightarrow} \vee$$

Figure 1.5: Extracting contexts schemata

Note also that $\Gamma \vdash \Gamma'; \mathcal{D}' \in S' \| S$ iff $\Gamma \vdash \Gamma'; \mathcal{D}' \in S \| S'$.

Lemma 1.11 (Schema Weakening) *Let* $\Gamma; \mathcal{D} < \mathcal{S}$. *Then*

1. *If* $\models_{\mathcal{S}} D$, *then* $\models_{\mathcal{S}\bar{\mathcal{S}}'} D$.
2. *If* $\Gamma; \mathcal{D} \setminus G < \mathcal{S}$, *then* $\Gamma; \mathcal{D} \setminus G < \mathcal{S}\bar{\mathcal{S}}'$.
3. *If* $\Gamma; \mathcal{D} < \mathcal{S}$ *then* $\Gamma; \mathcal{D} < \mathcal{S} \| \mathcal{S}'$.

Proof: By mutual induction on the structure of $\pi :: \models_{\mathcal{S}} D$, $\gamma :: \Gamma; \mathcal{D} \setminus G < \mathcal{S}$ and $\sigma :: \Gamma; \mathcal{D} < \mathcal{S}$. We show the crucial cases:

Case:

$$\pi = \frac{X \in \{Q, \top, \bot\}}{\models_{\mathcal{S}} X} \models_{\mathcal{S}} X$$

Trivially

$$\pi' = \frac{X \in \{Q, \top, \bot\}}{\models_{\mathcal{S}\bar{\mathcal{S}}'} Q} \models_{\mathcal{S}} X$$

Case:

$$\pi = \frac{\models_{\mathcal{S}} D \qquad \cdot; \top \setminus G < \mathcal{S}}{\models_{\mathcal{S}} G \to D} \models_{\mathcal{S}}\to$$

$\models_{\mathcal{S}} D$	By sub-derivation
$\models_{\mathcal{S}\bar{\mathcal{S}}'} D$	By IH 1
$\cdot; \top \setminus G < \mathcal{S}$	By sub-derivation
$\cdot; \top \setminus G < \mathcal{S}\bar{\mathcal{S}}'$	By IH 2
$\models_{\mathcal{S}\bar{\mathcal{S}}'} G \to D$	By rule $\models_{\mathcal{S}}\to$

Case:

$$\gamma = \frac{X \in \{Q, \top, \bot\} \text{ or (dis)eq} \qquad \Gamma; \mathcal{D} < \mathcal{S}}{\Gamma; \mathcal{D} \setminus X < \mathcal{S}} \setminus X$$

$\Gamma; \mathcal{D} < \mathcal{S}$	By sub-derivation
$\Gamma; \mathcal{D} < \mathcal{S}\bar{\mathcal{S}}'$	By IH 3
$\Gamma; \mathcal{D} \setminus X < \mathcal{S}\bar{\mathcal{S}}'$	By rule $\setminus X$

Case:

$$\gamma = \frac{\models_{\mathcal{S}} D \qquad \Gamma; \mathcal{D} \wedge D \setminus G < \mathcal{S}}{\Gamma; \mathcal{D} \setminus D \to G < \mathcal{S}} \setminus \to$$

$\models_{\mathcal{S}} D$	By sub-derivation
$\models_{\mathcal{S}\bar{\mathcal{S}}'} D$	By IH 1
$\Gamma; \mathcal{D} \wedge D \setminus G < \mathcal{S}$	By sub-derivation
$\Gamma; (\mathcal{D} \wedge D) \setminus G < \mathcal{S}\bar{\mathcal{S}}'$	By IH 2
$\Gamma; \mathcal{D} \setminus D \to G < \mathcal{S}\bar{\mathcal{S}}'$	By rule $\setminus \to$

Case:

$$\sigma = \frac{}{\cdot; \top < \mathcal{S}} <_1$$

Then immediately:

$$\sigma' = \frac{}{\cdot; \top < \mathcal{S} \| \mathcal{S}'} <_1$$

Case:

$$\sigma = \frac{\Gamma \vdash (\Gamma';\mathcal{D}') \in \mathcal{S} \quad \models_{\mathcal{S}} \mathcal{D}' \quad (\Gamma;\mathcal{D}) < \mathcal{S}}{(\Gamma, \lceil\Gamma'\rceil);(\mathcal{D} \wedge \lceil\mathcal{D}'\rceil) < \mathcal{S}} <_2$$

$\Gamma;\mathcal{D} < \mathcal{S}$	By sub-derivation
$\Gamma;\mathcal{D} < \mathcal{S}\bar{\mathcal{S}}'$	By IH 3
$\Gamma \vdash \Gamma';\mathcal{D}' \in \mathcal{S}$	By sub-derivation
$\Gamma \vdash \Gamma';\mathcal{D}' \in \mathcal{S}\vert\vert\mathcal{S}'$	By Lemma 6.10
$\models_{\mathcal{S}} \mathcal{D}'$	By sub-derivation
$\models_{\mathcal{S}\bar{\mathcal{S}}'} D$	By IH 1
$(\Gamma, \lceil\Gamma'\rceil);(\mathcal{D} \wedge \lceil\mathcal{D}'\rceil) < \mathcal{S}\bar{\mathcal{S}}'$	By rule $<_2$

□

We are now ready to show that if a program yields a schema, then every instance of a clause used at run-rime will comply with the schema. We need to generalize this statement to take into account instances of goals as well; in the latter case, we need to synchronize the schema accumulator in $\Gamma';\mathcal{D}' \vdash G \stackrel{G}{\Longrightarrow} \mathcal{S}$ with the possibly incomplete run-time context $\Gamma;\mathcal{D}$. A fortiori the result holds for compile-time contexts as well, by taking $\Gamma;\mathcal{D}$ as $\cdot;\top$.

Theorem 1.12 (Extracted Schema Satisfaction) *Let* $\Gamma \vdash \theta : \Phi$.

1. *If* $\vdash_\Phi D \stackrel{D}{\Longrightarrow} \mathcal{S}$, *then* $\models_{\mathcal{S}} [\theta]D$.
2. *If* $\Gamma';\mathcal{D}' \vdash_\Phi G \stackrel{G}{\Longrightarrow} \mathcal{S}$, $\models_{\mathcal{S}} \mathcal{D}'$ *and* $\Gamma;\mathcal{D} < \mathcal{S}$, *then* $(\Gamma,\Gamma');(\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]G < \mathcal{S}$.

Proof: By mutual induction on the structure of $\pi :: D \stackrel{D}{\Longrightarrow} \mathcal{S}$ and $\gamma :: \Gamma';\mathcal{D}' \vdash_\Phi G \stackrel{G}{\Longrightarrow} \mathcal{S}$.

Case:

$$\pi = \frac{X \in \{Q, \top, \bot\}}{\vdash_\Phi X \stackrel{D}{\Longrightarrow} \cdot;\top} \stackrel{D}{\Longrightarrow} X$$

$\models_{\cdot;\top} X$	By rule $\models_{\mathcal{S}} X$

Case:

$$\pi = \frac{\vdash_\Phi D_1 \stackrel{D}{\Longrightarrow} \mathcal{S}_1 \qquad \vdash_\Phi D_2 \stackrel{D}{\Longrightarrow} \mathcal{S}_2}{\vdash_\Phi D_1 \wedge D_2 \stackrel{D}{\Longrightarrow} \mathcal{S}_1\vert\vert\mathcal{S}_2} \stackrel{D}{\Longrightarrow} \wedge$$

$\vdash_\Phi D_1 \stackrel{D}{\Longrightarrow} \mathcal{S}_1$	By sub-derivation
$\vdash_\Phi D_2 \stackrel{D}{\Longrightarrow} \mathcal{S}_2$	By sub-derivation
$\models_{\mathcal{S}_1} [\theta]D_1$	By IH 1
$\models_{\mathcal{S}_2} [\theta]D_2$	By IH 1
$\models_{\mathcal{S}_1\vert\vert\mathcal{S}_2} [\theta]D_1$	By Lemma 6.11
$\models_{\mathcal{S}_1\vert\vert\mathcal{S}_2} [\theta]D_2$	By Lemma 6.11
$\models_{\mathcal{S}_1\vert\vert\mathcal{S}_2} [\theta]D_1 \wedge [\theta]D_2$	By rule $\models_{\mathcal{S}} \wedge$
$\models_{\mathcal{S}_1\vert\vert\mathcal{S}_2} [\theta](D_1 \wedge D_2)$	By subst.

Case: π ends in $\stackrel{D}{\Longrightarrow} \vee$: analogously to the above.

Case: π ends in $\stackrel{D}{\Longrightarrow} \forall^y$: by an immediate appeal to IH 1.

Case:

$$\pi = \frac{\cdot; \top \vdash_\Phi G \overset{G}{\Longrightarrow} \mathcal{S}_1 \qquad \vdash_\Phi D \overset{D}{\Longrightarrow} \mathcal{S}_2}{\vdash_\Phi (G \to D) \overset{D}{\Longrightarrow} \mathcal{S}_1 \| \mathcal{S}_2} \overset{D}{\Longrightarrow}\to$$

$\vdash_\Phi D \overset{D}{\Longrightarrow} \mathcal{S}_2$	By sub-derivation
$\models_{\mathcal{S}_2} [\theta]D$	By IH 1
$\models_{\mathcal{S}_1\|\mathcal{S}_2} [\theta]D$	By Lemma 6.11
$\cdot; \top \vdash_\Phi G \overset{G}{\Longrightarrow} \mathcal{S}_1$	By sub-derivation
$\cdot; \top < \mathcal{S}_1$	By rule $<_1$
$\cdot; \top \setminus [\theta]G < \mathcal{S}_1$	By IH 2
$\cdot; \top \setminus [\theta]G < \mathcal{S}_1\|\mathcal{S}_2$	By Lemma 6.11
$\models_{\mathcal{S}_1\|\mathcal{S}_2} [\theta]G \to [\theta]D$	By rule $\models_{\mathcal{S}}\to$

Case:

$$\gamma = \frac{X \in \{Q, \top, \bot\} \text{ or (dis)eq}}{\Gamma'; \mathcal{D}' \vdash_\Phi X \overset{G}{\Longrightarrow} \Gamma'; \text{SOME } \Phi_{\mathcal{D}'}. \mathcal{D}'} \overset{G}{\Longrightarrow} X$$

$\Gamma'; [\theta]\mathcal{D}' \equiv_\alpha \Gamma'; [\theta]\mathcal{D}'$	By hypothesis
$\Gamma \vdash \Gamma'; [\theta]\mathcal{D}' \in \Gamma'; \text{SOME } \Phi_{\mathcal{D}'}. \mathcal{D}'$	By rule $\in_1$
$\Gamma; \mathcal{D} < \Gamma'; \text{SOME } \Phi_{\mathcal{D}'}. \mathcal{D}'$	By hypothesis
$\models_{\Gamma';(\text{SOME } \Phi_{\mathcal{D}'}. \mathcal{D})} \mathcal{D}'$	By hypothesis
$\models_{\Gamma';(\text{SOME } \Phi_{\mathcal{D}'}. \mathcal{D})} [\theta]\mathcal{D}'$	By Lemma 6.8
$(\Gamma, \lceil\Gamma'\rceil); (\mathcal{D} \wedge \lceil[\theta]\mathcal{D}'\rceil) < \Gamma'; \text{SOME } \Phi_{\mathcal{D}'}. \mathcal{D}'$	By rule $<_2$
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]X < \Gamma'; \text{SOME } \Phi_{\mathcal{D}'}. \mathcal{D}'$	By rule $\setminus X$

Case:

$$\gamma = \frac{(\Gamma', y{:}A); \mathcal{D}' \vdash_\Phi [y/x]G \overset{G}{\Longrightarrow} \mathcal{S}}{\Gamma'; \mathcal{D}' \vdash_\Phi \forall x{:}A.\, G \overset{G}{\Longrightarrow} \mathcal{S}} \overset{G}{\Longrightarrow} \forall^y$$

$(\Gamma', y{:}A); \mathcal{D}' \vdash_\Phi [y/x]G \overset{G}{\Longrightarrow} \mathcal{S}$	By sub-derivation
$(\Gamma, \Gamma', y{:}A); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta] \cdot [y/x]G < \mathcal{S}$	By IH 2
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]\forall x{:}A.\, G < \mathcal{S}$	By rule $\setminus\forall^y$

Case:

$$\gamma = \frac{\Gamma'; (\mathcal{D}' \wedge D) \vdash_\Phi G \overset{G}{\Longrightarrow} \mathcal{S}_1 \qquad \vdash_\Phi D \overset{D}{\Longrightarrow} \mathcal{S}_2}{\Gamma'; \mathcal{D}' \vdash_\Phi D \to G \overset{G}{\Longrightarrow} \mathcal{S}_1 \| \mathcal{S}_2} \overset{G}{\Longrightarrow}\to$$

$\vdash_\Phi D \overset{D}{\Longrightarrow} \mathcal{S}_2$	By sub-derivation
$\models_{\mathcal{S}_2} [\theta]D$	By IH 1
$\models_{\mathcal{S}_1\|\mathcal{S}_2} [\theta]D$	By Lemma 6.11
$\Gamma; \mathcal{D} < \mathcal{S}_1$	By hypothesis
$\Gamma'; (\mathcal{D}' \wedge D) \vdash_\Phi G \overset{G}{\Longrightarrow} \mathcal{S}_1$	By sub-derivation
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}' \wedge [\theta]D) \setminus [\theta]G < \mathcal{S}_1$	By IH 2
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}' \wedge [\theta]D) \setminus [\theta]G < \mathcal{S}_1\|\mathcal{S}_2$	By Lemma 6.11
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]D \to [\theta]G < \mathcal{S}_1\|\mathcal{S}_2$	By rule $\setminus\to$

Case:

$$\gamma = \frac{\Gamma'; \mathcal{D}' \vdash_\Phi G_1 \overset{G}{\Longrightarrow} \mathcal{S}_1 \qquad \Gamma'; \mathcal{D}' \vdash_\Phi G_2 \overset{G}{\Longrightarrow} \mathcal{S}_2}{\Gamma'; \mathcal{D}' \vdash_\Phi G_1 \wedge G_2 \overset{G}{\Longrightarrow} \mathcal{S}_1 \| \mathcal{S}_2} \overset{G}{\Longrightarrow} \wedge$$

$\Gamma'; \mathcal{D}' \vdash_\Phi G_1 \stackrel{G}{\Longrightarrow} \mathcal{S}_1$ By sub-derivation
$\Gamma; \mathcal{D} < \mathcal{S}_1$ By hypothesis
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]G_1 < \mathcal{S}_1$ By IH 2
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]G_1 < \mathcal{S}_1 \| \mathcal{S}_2$ By Lemma 6.11
$\Gamma'; \mathcal{D}' \vdash_\Phi G_2 \stackrel{G}{\Longrightarrow} \mathcal{S}_2$ By sub-derivation
$\Gamma; \mathcal{D} < \mathcal{S}_2$ By hypothesis
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]G_2 < \mathcal{S}_2$ By IH 2
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]G_2 < \mathcal{S}_1 \| \mathcal{S}_2$ By Lemma 6.11
$(\Gamma, \Gamma'); (\mathcal{D} \wedge [\theta]\mathcal{D}') \setminus [\theta]G_1 \wedge [\theta]G_2 < \mathcal{S}_1 \| \mathcal{S}_2$ By rule $\setminus\wedge$

Case: γ ends in $\stackrel{G}{\Longrightarrow} \vee$: similarly to the above. to IH 2.

□

1.2.2 Context Preservation

We aim to prove that execution preserves contexts, provided that the program itself and the input goal satisfy a schema; that is, that every subgoal which arises in any given successful or failed (immediate and non-immediate sub-derivation) satisfies the same context schema.

Note that we have already established schema extraction (Theorem 6.12), that is we assume that $\mathcal{P} \stackrel{D}{\Longrightarrow} \mathcal{S}$. We write $\pi' < \pi$ to say that π' is a strict subproof of π, i.e. $\pi' \leq \pi$ if $\pi' < \pi$ or $\pi' = \pi$.

Theorem 1.13 (Context Preservation w.r.t. Provability) *Let $\models_\mathcal{S} \mathcal{P}$ and $D \sqsubseteq (\mathcal{P} \wedge \mathcal{D})$. For any $\Gamma; \mathcal{D} < \mathcal{S}$, if ($\pi :: \Gamma; \mathcal{D} \vdash_\mathcal{P} G$ and $\Gamma; \mathcal{D} \setminus G < \mathcal{S}$) or ($\iota :: \Gamma; \mathcal{D} \vdash_\mathcal{P} D \gg Q$ and $\models_\mathcal{S} D$), then:*

1. *for every subproof $\pi' :: \Gamma'; \mathcal{D}' \vdash_\mathcal{P} G'$ and $\Gamma'; \mathcal{D}' \setminus G' < \mathcal{S}$.*
2. *for every subproof $\iota' :: \Gamma'; \mathcal{D}' \vdash_\mathcal{P} D' \gg Q$ and $\models_\mathcal{S} D'$.*

Proof: Bymutual induction on the structure of $\pi :: \Gamma; \mathcal{D} \vdash_\mathcal{P} G$ and $\iota :: \Gamma; \mathcal{D} \vdash_\mathcal{P} D \gg Q$.

Case: $\pi = \Gamma; \mathcal{D} \vdash_\mathcal{P} \top$: trivial.

Case:

$$\pi = \frac{\begin{array}{c}\pi_1\\ \Gamma; \mathcal{D} \wedge D \vdash_\mathcal{P} G\end{array}}{\Gamma; \mathcal{D} \vdash_\mathcal{P} D \to G} \vdash\to$$

Subcase: $\pi' = \pi$: trivial.

Subcase: $\pi' < \pi$: then $\pi' \leq \pi_1$.

$\pi_1 :: \Gamma; \mathcal{D} \wedge D \vdash_\mathcal{P} G$ By sub-derivation
$\Gamma; \mathcal{D} \setminus D \to G < \mathcal{S}$ By hypothesis
$\Gamma; \mathcal{D} \wedge D \setminus G < \mathcal{S}$ By inversion
$\iota' :: \Gamma'; \mathcal{D}' \vdash_\mathcal{P} D' \gg Q$ and $\models_\mathcal{S} D'$ and
$\pi_1' :: \Gamma'; (\mathcal{D}' \wedge D') \vdash_\mathcal{P} G'$ and $\Gamma'; (\mathcal{D}' \wedge D') \setminus G' < \mathcal{S}$ By IH
$\pi' :: \Gamma'; \mathcal{D}' \vdash_\mathcal{P} D' \to G'$ and $\Gamma'; \mathcal{D}' \setminus D' \to G' < \mathcal{S}$ By rule

Case:

$$\pi \frac{\begin{array}{c}\pi_1\\ (\Gamma, y{:}A); \mathcal{D} \vdash_\mathcal{P} [y/x]G\end{array}}{\Gamma; \mathcal{D} \vdash_\mathcal{P} \forall x{:}A.\, G} \vdash \forall^y$$

Subcase: $\pi' = \pi$: trivial.

Subcase: $\pi' < \pi$: then $\pi' \leq \pi_1$.

$\pi_1 :: (\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} [y/x]G$ By sub-derivation
$\Gamma; \mathcal{D} \setminus \forall x{:}A.\, G < \mathcal{S}$ By hypothesis
$(\Gamma, y{:}A); \mathcal{D} \setminus [y/x]G < \mathcal{S}$ By inversion
$\iota' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} D' \gg Q$ and $\models_{\mathcal{S}} D'$ and
$\pi_1' :: (\Gamma', y{:}A); \mathcal{D}' \vdash_{\mathcal{P}} [y/x]G'$ and $(\Gamma', y{:}A); \mathcal{D}' \setminus [y/x]G' < \mathcal{S}$ By IH
$\pi' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} \forall x{:}A.\, G'$ and $\Gamma'; \mathcal{D}' \setminus \forall x{:}A.\, G' < \mathcal{S}$ By rule

Case: π ends in $\vdash \wedge$, $\vdash \vee_1$, $\vdash \vee_2$: similarly.

Case:

$$\pi = \frac{\overset{\iota}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q}}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} Q} \vdash \mathrm{At}$$

Subcase: $\pi' = \pi$: trivial.

Subcase: $\pi' < \pi$: then $\pi' \leq \iota$.

Subcase: $\iota_1 :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q$, for $D \sqsubseteq \mathcal{P}$ By sub-derivation
$\models_{\mathcal{S}} D$ By hypothesis
$\pi' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} G'$ and $\Gamma'; \mathcal{D}' \setminus G' < \mathcal{S}$ and
$\iota_1' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} D' \gg Q$ and $\models_{\mathcal{S}} D'$ By IH
$\Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} Q'$ and $\Gamma'; \mathcal{D}' \setminus Q' < \mathcal{S}$ By rule

Subcase: $\iota_1 :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q$, for $D \sqsubseteq \mathcal{D}$ By sub-derivation
$\models_{\mathcal{S}} D$ By Lemma 6.7
$\pi' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} G'$ and $\Gamma'; \mathcal{D}' \setminus G' < \mathcal{S}$ and
$\iota_1' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} D' \gg Q$ and $\models_{\mathcal{S}} D'$ By IH
$\Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} Q'$ and $\Gamma'; \mathcal{D}' \setminus Q' < \mathcal{S}$ By rule

Case: $\iota :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bot \gg Q$: trivial.

Case:

$$\iota = \frac{\overset{\iota_1}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q}}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \wedge D_2 \gg Q} \gg\wedge_1$$

Subcase: $\iota' = \iota$: trivial.

Subcase: $\iota' < \iota$: then $\iota' \leq \iota_1$:

$\models_{\mathcal{S}} D_1 \wedge D_2$ By hypothesis
$\models_{\mathcal{S}} D_1$ and $\models_{\mathcal{S}} D_2$ By inversion
$\iota_1 :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$ By sub-derivation
$\pi' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} G'$ and $\Gamma'; \mathcal{D}' \setminus G' < \mathcal{S}$ and
$\iota_1' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} D_1' \gg Q$ and $\models_{\mathcal{S}} D_1'$ By IH
$\iota' :: \Gamma'; \mathcal{D}' \vdash_{\mathcal{P}} D_1' \wedge D_2' \gg Q$ and $\models_{\mathcal{S}} D_1' \wedge D_2'$ By rule

Case: ι ends in $\gg\wedge_2$. Symmetrical.

Case:

$$\iota = \frac{\overset{\iota_1}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [t/x]D \gg Q}}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall x{:}A.\, D \gg Q} \gg\forall$$

Subcase: $\iota' = \iota$: trivial.

Subcase: $\iota' < \iota$: then $\iota' \leq \iota_1$.

$\iota_1 :: \Gamma;\mathcal{D} \vdash_{\mathcal{P}} [t/x]D \gg Q$	By sub-derivation
$\models_{\mathcal{S}} \forall x{:}A.\, D$	By hypothesis
$\models_{\mathcal{S}} [u/x]D$	By inversion
$\models_{\mathcal{S}} [t/u, u/x]D$	By substitution
$\pi' :: \Gamma';\mathcal{D}' \vdash_{\mathcal{P}} G'$ and $\Gamma';\mathcal{D}' \setminus G' < \mathcal{S}$ and	
$\iota_1' :: \Gamma';\mathcal{D}' \vdash_{\mathcal{P}} [t/x]D' \gg Q$ and $\models_{\mathcal{S}} [t/x]D'$	By IH
$\iota' :: \Gamma';\mathcal{D}' \vdash_{\mathcal{P}} \forall x{:}A.\, D' \gg Q$ and $\models_{\mathcal{S}} \forall x{:}A.\, D'$	By rule

Case:

$$\iota = \cfrac{\overset{\iota_1}{\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D \gg Q} \qquad \overset{\pi_1}{\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G}}{\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G \to D \gg Q} \gg\to$$

Subcase: $\iota' = \iota$: trivial.

Subcase: $\iota' < \iota$:

Subcase: $\iota' \leq \iota_1$

$\iota_1 :: \Gamma;\mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ and $\pi_1 :: \Gamma;\mathcal{D} \vdash_{\mathcal{P}} G$	By sub-derivation
$\models_{\mathcal{S}} G \to D$	By hypothesis
$\Gamma;\mathcal{D} \setminus G < \mathcal{S}$ and $\models_{\mathcal{S}} D$	By inversion
$\pi' :: \Gamma';\mathcal{D}' \vdash_{\mathcal{P}} G'$ and $\Gamma';\mathcal{D}' \setminus G' < \mathcal{S}$ and	
$\iota_1' :: \Gamma';\mathcal{D}' \vdash_{\mathcal{P}} D' \gg Q$ and $\models_{\mathcal{S}} D'$	By IH
$\iota' :: \Gamma';\mathcal{D}' \vdash_{\mathcal{P}} G' \to D' \gg Q$ and $\models_{\mathcal{S}} G' \to D'$	By rule

Subcase: $\iota' \leq \pi_1$: analogously.

Case: ι ends in $\gg\vee, \gg\forall$: similarly.

□

We establish an analogous result w.r.t. denial.

Theorem 1.14 (Context Preservation w.r.t. Denial) *Let* $\models_{\mathcal{S}} \mathcal{P}$ *and* $D \sqsubseteq (\mathcal{P} \wedge \mathcal{D})$*: for any* $\Gamma;\mathcal{D} < \mathcal{S}$*, if (* $\pi :: \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G$ *and* $\Gamma;\mathcal{D} \setminus G < \mathcal{S}$ *) or (* $\iota :: \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D \gg Q$ *and* $\models_{\mathcal{S}} D$*), then:*

1. *for every subproof* $\pi' :: \Gamma';\mathcal{D}' \nvdash_{\mathcal{P}} G'$ *and* $\Gamma';\mathcal{D}' \setminus G' < \mathcal{S}$*.*
2. *for every subproof* $\iota' :: \Gamma';\mathcal{D}' \nvdash_{\mathcal{P}} D' \gg Q$ *and* $\models_{\mathcal{S}} D'$*.*

Proof: By mutual induction on the structure of $\pi :: \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G$ and $\iota :: \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D \gg Q$, similarly to the case of provability. We show only some cases:

Case:

$$\pi = \cfrac{\overset{\pi_1}{\Gamma;(\mathcal{D} \wedge D) \nvdash_{\mathcal{P}} G}}{\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D \to G} \nvdash\to$$

Subcase: $\pi' = \pi$: trivial.

Subcase: $\pi' < \pi$: then $\pi' \leq \pi_1$:

$\Gamma;(\mathcal{D} \wedge D) \nvdash_{\mathcal{P}} G$	By sub-derivation
$\Gamma;\mathcal{D} \setminus D \to G < \mathcal{S}$	By hypothesis
$\Gamma;\mathcal{D} \wedge D \setminus G < \mathcal{S}$	By inversion
$\iota' :: \Gamma';\mathcal{D}' \nvdash_{\mathcal{P}} D' \gg Q$ and $\models_{\mathcal{S}} D'$ and	
$\pi' :: \Gamma;(\mathcal{D}' \wedge D') \nvdash_{\mathcal{P}} G'$ and $\Gamma;(\mathcal{D}' \wedge D') \setminus G' < \mathcal{S}$	By IH
$\pi'' :: \Gamma';\mathcal{D}' \nvdash_{\mathcal{P}} (D' \to G')$ and $\Gamma';\mathcal{D}' \setminus (D' \to G') < \mathcal{S}$	By rule

Case:

$$\iota = \cfrac{\overset{\iota_1}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G}}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \leftarrow G \gg Q} \not\gg\!\!\to_1$$

Subcase: $\iota' = \iota$: trivial.

Subcase: $\iota' < \iota$: then $\iota' \leq \iota_1$:

$\iota_1 :: \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G$	By sub-derivation
$\models_{\mathcal{S}} G \to D$	By hypothesis
$\Gamma; \mathcal{D} \setminus G < \mathcal{S}$ and $\models_{\mathcal{S}} D$	By inversion
$\pi' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} G'$ and $\Gamma'; \mathcal{D}' \setminus G' < \mathcal{S}$ and	
$\iota' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} D' \gg Q$ and $\models_{\mathcal{S}} D'$	By IH
$\iota'' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} (D' \leftarrow G') \gg Q$ and $\models_{\mathcal{S}} (D' \leftarrow G')$	By rule

Case:

$$\iota = \cfrac{\overset{\iota_1}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \gg Q}}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \leftarrow G \gg Q} \not\gg\!\!\to_2$$

Subcase: $\iota' = \iota$: trivial.

Subcase: $\iota' < \iota$: then $\iota' \leq \iota_1$:

$\iota_1 :: \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \gg Q$	By sub-derivation
$\models_{\mathcal{S}} G \to D$	By hypothesis
$\Gamma; \mathcal{D} \setminus G < \mathcal{S}$ and $\models_{\mathcal{S}} D$	By inversion
$\pi' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} G'$ and $\Gamma'; \mathcal{D}' \setminus G' < \mathcal{S}$ and	
$\iota' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} D' \gg Q$ and $\models_{\mathcal{S}} D'$	By IH
$\iota'' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} (D' \leftarrow G') \gg Q$ and $\models_{\mathcal{S}} (D' \leftarrow G')$	By rule

Case:

$$\iota = \cfrac{\overset{\iota_1}{\text{for all } n\ \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [n/x] D \gg Q}}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \forall x\!:\!A.\, D \gg Q} \gg\!\forall$$

Subcase: $\iota' = \iota$: trivial.

Subcase: $\iota' < \iota$: then $\iota' \leq \iota_1$.

$\models_{\mathcal{S}} \forall x\!:\!A.\, D$	By hypothesis
$\models_{\mathcal{S}} [u/x] D$	By inversion
For all $\iota_1 :: \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [n/x] D \gg Q$	By sub-derivation
For all $\models_{\mathcal{S}} [n/u, u/x] D$	By substitution
$\pi' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} G'$ and $\Gamma'; \mathcal{D}' \setminus G' < \mathcal{S}$ and	
For all $\iota_1' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} [n/x] D' \gg Q$ and $\models_{\mathcal{S}} [n/x] D'$	By IH
$\iota' :: \Gamma'; \mathcal{D}' \not\vdash_{\mathcal{P}} \forall x\!:\!A.\, D' \gg Q$ and $\models_{\mathcal{S}} \forall x\!:\!A.\, D'$	By rule

□

1.3 Terminating Programs

We now introduce terminating programs. The notion of termination that we adopt is the very strong *universal* one, as it is known in logic programming [?]. This is abstractly achieved by means of a relation between goals or between goals and clause heads, namely '$\prec$', with the intended meaning of "G_1 $[G]$ can arise as a subgoal of G_2 $[D]$". Intuitively, a program terminates if it yields an ordering relation which admits no infinite descending chains. This is defined in Figure 6.6. We do not commit here to actual ways to verify the latter property. This issue is explored, in the higher-order setting, for example in [?]. More formally:

$$\mathit{Ordering} \quad R \quad ::= \quad \cdot \mid R, G \prec D \mid R, G_1 \prec G_2$$

Let $\mathcal{P} \overset{D}{\Longrightarrow} R$ and take the stable (that is, closed under substitutions) and transitive closure of R, call it $[R]$. We say that $\mathcal{P}$ is *terminating*, denoted $\mathcal{P}\downarrow$, whenever $[R]$ is well-founded.

Example 1.15 *If* even *defines the even number,* **even**$\overset{D}{\Longrightarrow}$ $even(X) \prec even(s(s(X)))$*. Furthermore the closure of this relation is well-founded as it can be proven say by induction on X. Consider instead the following:*

$$\begin{array}{l} \forall E\!:\!exp.\forall M\!:\!nat. \\ p\ (lam\ E)\ M \\ \quad \leftarrow (\forall x\!:\!exp.(\forall N\!:\!nat.\, p\ x\ (s\ N) \leftarrow p\ x\ (s\ N)) \rightarrow p\ (E\ x)\ M). \end{array}$$

The clause is not terminating as the subgoal relation contains the pair $[y/x](p\ x\ s(N) \prec p\ x\ s(N))$, which yields an infinite descending chain $(p\ y\ (s\ 0)) \prec (p\ y\ (s\ 0)) \prec \ldots$.

Finally, we define when a *schema* is terminating:

$$\frac{}{\circ\downarrow}\circ\downarrow \qquad \frac{\Gamma \vdash \mathcal{D}\downarrow \qquad \mathcal{S}\downarrow}{\mathcal{S}\|\Gamma;\mathrm{SOME}\ \Phi.\mathcal{D}\downarrow}\|\downarrow$$

Since Φ ranges over $FV(\mathcal{D}) \setminus \Gamma$, we verify the termination of $\mathcal{D}$ in the Γ context.

Lemma 1.16

1. *If $\Gamma \vdash \mathcal{D} \overset{D}{\Longrightarrow} R$ and $D \sqsubseteq \mathcal{D}$ such that $\Gamma \vdash D \overset{D}{\Longrightarrow} R'$, then $R' \subseteq R$.*
2. *If $\Gamma \vdash G \overset{G}{\Longrightarrow} R$, $G' \prec G$ and $\Gamma \vdash G' \overset{G}{\Longrightarrow} R'$, then $R' \subseteq R$.*

Proof: By a straightforward mutual induction on the structure of $\Gamma \vdash D \overset{D}{\Longrightarrow} R'$ and $\Gamma \vdash G' \overset{G}{\Longrightarrow} R'$. □

Corollary 1.17 *If $\Gamma \vdash \mathcal{D}\downarrow$ and $D \sqsubseteq \mathcal{D}$, then $D\downarrow$.*

Proof: By definition, $\Gamma \vdash \mathcal{D}\downarrow$ if $\Gamma \vdash \mathcal{D} \overset{D}{\Longrightarrow} R$ and $[R]$ is well-founded; by Lemma 6.16 $\Gamma \vdash D \overset{D}{\Longrightarrow} R'$, and $R' \subseteq R$. Thus $[R']$ is well-founded, and $D\downarrow$. □

Lemma 1.18 *Let $\mathcal{S}\downarrow$. If $\Gamma;\mathcal{D} < \mathcal{S}$ and $D \sqsubseteq \mathcal{D}$, then $D\downarrow$.*

Proof: By induction on the structure of $\mathcal{S}\downarrow$, using Corollary 6.17. □

We can now prove that if a program is terminating, a non-proof of any ground G is a *denial* of G; we are going to reason classically that either there is a proof of G, or there is not such a proof. We recall that $\not\vdash_{\mathcal{P}}$ is the denial relation introduced in Figure 6.1 and 6.2.

Theorem 1.19 (Termination) *Let $\mathcal{P}\downarrow$ and $\mathcal{S}\downarrow$ be a schema such that $\models_{\mathcal{S}} \mathcal{P}$ and $\Gamma;\mathcal{D} < \mathcal{S}$: for any ground G, if not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G$, then $\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} G$.*

Proof: We generalize this to:

1. If not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G$, then $\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} G$.
2. For every $D \sqsubseteq \mathcal{P} \wedge \mathcal{D}$, if not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D \gg Q$, then $\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} D \gg Q$.

We proceed by mutual induction on the goal ordering induced by $\mathcal{P}$ terminating and on the structure of D: we start with 2. Since D is an instance of a terminating clause either from the program or the run-time context, by stability and Corollary 6.18, it is terminating as well.

$$\frac{X \in \{Q, \top, \bot\} \text{ or (dis)eq}}{\Gamma \vdash X \stackrel{G}{\Longrightarrow} \cdot} \stackrel{G}{\Longrightarrow} X$$

$$\frac{\Gamma \vdash G \stackrel{G}{\Longrightarrow} R_1 \qquad \Gamma \vdash D \stackrel{D}{\Longrightarrow} R_2}{\Gamma \vdash D \to G \stackrel{G}{\Longrightarrow} R_1, R_2, G \prec D \to G} \stackrel{G}{\Longrightarrow}\to$$

$$\frac{\Gamma, y{:}A \vdash G \stackrel{G}{\Longrightarrow} R}{\Gamma \vdash \forall x : A.\, G \stackrel{G}{\Longrightarrow} R, [y/x]G \prec \forall x : A.\, G} \stackrel{G}{\Longrightarrow} \forall^y$$

$$\frac{\Gamma \vdash G_1 \stackrel{G}{\Longrightarrow} R_1 \qquad \Gamma \vdash G_2 \stackrel{G}{\Longrightarrow} R_2}{\Gamma \vdash G_1 \wedge G_2 \stackrel{G}{\Longrightarrow} R_1, R_2, G_1 \prec G_1 \wedge G_2, G_2 \prec G_1 \wedge G_2} \stackrel{G}{\Longrightarrow} \wedge$$

$$\frac{\Gamma \vdash G_1 \stackrel{G}{\Longrightarrow} R_1 \qquad \Gamma \vdash G_2 \stackrel{G}{\Longrightarrow} R_2}{\Gamma \vdash G_1 \vee G_2 \stackrel{G}{\Longrightarrow} R_1, R_2, G_1 \prec G_1 \vee G_2, G_2 \prec G_1 \vee G_2,} \stackrel{G}{\Longrightarrow} \vee$$

$$\frac{X \in \{Q, \top, \bot\}}{\Gamma \vdash X \stackrel{D}{\Longrightarrow} \cdot} \stackrel{D}{\Longrightarrow} X$$

$$\frac{\Gamma \vdash D \stackrel{D}{\Longrightarrow} R_1 \qquad \Gamma \vdash G \stackrel{G}{\Longrightarrow} R_2}{\Gamma \vdash (G \to D) \stackrel{D}{\Longrightarrow} R_1, R_2, G \prec D} \stackrel{D}{\Longrightarrow}\to$$

$$\frac{\Gamma \vdash [u/x]D \stackrel{D}{\Longrightarrow} R}{\Gamma \vdash \forall x : A.\, D \stackrel{D}{\Longrightarrow} R} \stackrel{D}{\Longrightarrow} \forall^u$$

$$\frac{\Gamma \vdash D_1 \stackrel{D}{\Longrightarrow} R_1 \qquad \Gamma \vdash D_2 \stackrel{D}{\Longrightarrow} R_2}{\Gamma \vdash D_1 \wedge D_2 \stackrel{D}{\Longrightarrow} R_1, R_2} \stackrel{D}{\Longrightarrow} \wedge$$

$$\frac{\Gamma \vdash D_1 \stackrel{D}{\Longrightarrow} R_1 \qquad \Gamma \vdash D_2 \stackrel{D}{\Longrightarrow} R_2}{\Gamma \vdash D_1 \vee D_2 \stackrel{D}{\Longrightarrow} R_1, R_2} \stackrel{D}{\Longrightarrow} \vee$$

Figure 1.6: Generation of the subgoal relation

Case: $D = \bot$: trivial.

Case: $D = \top$.

$$\frac{}{\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} \top \gg Q} \gg\top$$

Case: $D = D_1 \leftarrow G_1$. Either $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$ or not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$ and $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$ or not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$. By termination of $D_1 \leftarrow G_1$, $G_1 \prec D_1$, hence we can apply the IH to G_1: There are four sub-cases:

1.	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$	Subcase
	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \leftarrow G_1 \gg Q$	By rule $\gg \rightarrow$
2.	Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$	Subcase
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} G_1$	By IH 1
	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \leftarrow G_1 \gg Q$	By rule $\gg \rightarrow_2$
3.	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$	Subcase
	Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \gg Q$	By IH 2
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \leftarrow G_1 \gg Q$	By rule $\gg \rightarrow_1$
4.	Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$	Subcase
	Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \gg Q$	By IH 2
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \leftarrow G_1 \gg Q$By rule	By rule $\gg \rightarrow_1$

Case: $D = D_1 \vee D_2$.

1.	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \vee D_2 \gg Q$	By rule $\gg\vee$
2.	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_2 \gg Q$	By IH 2
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \vee D_2 \gg Q$	By rule
3.	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	Subcase
	Not $\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \gg Q$	By IH 2
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \vee D_2 \gg Q$	By rule
4.	Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \gg Q$	By IH 2
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_2 \gg Q$	By IH 2
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_1 \vee D_2 \gg Q$	By rule

Case: $D = D_1 \wedge D_2$.

1.	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \wedge D_2 \gg Q$	By rule $\gg\wedge$
2.	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
	Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	Subcase
	$\Gamma; \mathcal{D} \nvdash_{\mathcal{P}} D_2 \gg Q$	By IH 2
	$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \wedge D_2 \gg Q$	By rule

3. $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	Subcase
Not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D_1 \gg Q$	By IH 2
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D_1 \wedge D_2 \gg Q$	By rule
4. Not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	Subcase
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D_1 \gg Q$	By IH 2
Not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	Subcase
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D_2 \gg Q$	By IH 2
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D_1 \wedge D_2 \gg Q$	By rule

Case: $D = \forall x : A.\, D'$. For some ground $\Gamma \vdash t : A$, $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} [t/x]D' \gg Q$ or for all ground t, not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} [t/x]D' \gg Q$:

1. $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} [t/x]D' \gg Q$	Subcase
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \forall x : A.\, D' \gg Q$	By rule $\gg\forall$
2. For all t ground not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} [t/x]D' \gg Q$	Subcase
For all t ground $\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} [t/x]D' \gg Q$	By IH 2
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} \forall x : A.\, D' \gg Q$	By rule

Case: $G = \top$: trivially true.

Case: $G = \bot$:

$$\frac{}{\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} \bot}\ \nvdash \bot$$

Case: $G = Q$: either $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ or for every D not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D \gg Q$:

1. $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D \gg Q$	Subcase
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} Q$	By rule
2. Not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ for every D	Subcase
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D \gg Q$	By IH 2
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} Q$	By rule

Case: $G = G_1 \wedge G_2$:

1. $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_1$	Subcase
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_2$	Subcase
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_1 \wedge G_2$	By rule
2. $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_1$	Subcase
Not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_2$	Subcase
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G_2$	By IH 1
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G_1 \wedge G_2$	By rule
3. $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_2$	Subcase
Not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_1$	Subcase
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G_1$	By IH 1
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G_1 \wedge G_2$	By rule
4. Not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_1$	Subcase
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G_1$	By IH 1
Not $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} G_2$	Subcase
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G_2$	By IH 1
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G_1 \wedge G_2$	By rule

Case: $G = G_1 \vee G_2$:

1. $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$ Subcase
   $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2$ Subcase
   $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1 \vee G_2$ By rule
2. $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$ Subcase
   Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2$ Subcase
   $\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_2$ By IH 1
   $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2 \vee G_1$ By rule
3. $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2$ Subcase
   Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$ Subcase
   $\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_1$ By IH 1
   $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2 \vee G_1$ By rule
4. Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$ Subcase
   $\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_1$ By IH 1
   Not $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2$ Subcase
   $\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_2$ By IH 1
   $\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_1 \vee G_2$ By rule

Case: $G = \forall x : A.\, G'$: for a new parameter y, $(\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} [y/x]G'$ or not $(\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} [y/x]G'$:

1. $(\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} [y/x]G'$ Subcase
   $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall x : A.\, G'$ By rule
2. Not $(\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} [y/x]G'$ Subcase
   $(\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} [y/x]G'$ By IH 1
   $\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \forall x : A.\, G'$ By rule

Case: $G = D' \to G'$:

1. $\Gamma; (\mathcal{D} \wedge D') \vdash_{\mathcal{P}} G'$ Subcase
   $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D' \to G'$ By rule
2. Not $\Gamma; (\mathcal{D} \wedge D') \vdash_{\mathcal{P}} G'$ Subcase
   $\Gamma; (\mathcal{D} \wedge D') \not\vdash_{\mathcal{P}} G'$ By IH 1
   $\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D' \to G'$ By rule

Case: The (dis)equality case follows immediately from the decidability of the $\doteq, \neq$ rules.

□

1.4 Complementable Clauses

We restrict ourselves to programs with:

- *Complementable* clauses as defined in Figure 6.7.
- Rules of the form $\forall(Q \leftarrow G)$, such that every (input) term in Q is rigid.
- Parameters of base type and occurring only in *head* position, called *Shallow Parameter Expressions*:

$$\mathit{SPE} \quad e_x \quad ::= \quad x : a \mid \lambda x \,.\, e_x$$

We do not formalize here the former assumptions on terms; note that the rigidity restriction applies only to predicates mutually recursive to non-Horn ones – see the comment at the end of the proof of Theorem 6.34.

$$\frac{X \in \{\top, \bot\} \text{ or (dis)eq}}{\Gamma; \mathcal{D} \vdash X \text{ compl}}\, cg\ X$$

$$\frac{}{\Gamma; \top \vdash Q \text{ compl}}\, cg\text{At}$$

$$\frac{\text{for all } D \sqsubseteq \mathcal{D} : \text{dom}(\Gamma) \cap par(D) \neq \emptyset}{\Gamma; \mathcal{D} \vdash Q \text{ compl}}\, cg\text{At}\cap$$

$$\frac{\Gamma; (\mathcal{D} \wedge D) \vdash G \text{ compl} \qquad \Gamma; \mathcal{D} \vdash D \text{ compl}}{\Gamma; \mathcal{D} \vdash D \to G \text{ compl}}\, cg \to$$

$$\frac{(\Gamma, y{:}a); \mathcal{D} \vdash [y/x]G \text{ compl}}{\Gamma; \mathcal{D} \vdash \forall x{:}a.\, G \text{ compl}}\, cg\forall^y$$

$$\frac{\Gamma; \mathcal{D} \vdash G_1 \text{ compl} \qquad \Gamma; \mathcal{D} \vdash G_2 \text{ compl}}{\Gamma; \mathcal{D} \vdash G_1 \wedge G_2 \text{ compl}}\, cg\wedge$$

$$\frac{\Gamma; \mathcal{D} \vdash G_1 \text{ compl} \qquad \Gamma; \mathcal{D} \vdash G_2 \text{ compl}}{\Gamma; \mathcal{D} \vdash G_1 \vee G_2 \text{ compl}}\, cg\vee$$

$$\frac{X \in \{\top, \bot\}}{\Gamma; \mathcal{D} \vdash X \text{ compl}}\, cd\ X$$

$$\frac{\Gamma; \mathcal{D} \vdash D \text{ compl} \qquad \Gamma; \mathcal{D} \vdash G \text{ compl}}{\Gamma; \mathcal{D} \vdash D \leftarrow G \text{ compl}}\, cd \leftarrow$$

$$\frac{\Gamma; \mathcal{D} \vdash [u/x]D \text{ compl}}{\Gamma; \mathcal{D} \vdash \forall x{:}A.\, D \text{ compl}}\, cd\forall^u$$

$$\frac{\Gamma; \mathcal{D} \vdash D_1 \text{ compl} \qquad \Gamma; \mathcal{D} \vdash D_2 \text{ compl}}{\Gamma; \mathcal{D} \vdash D_1 \wedge D_2 \text{ compl}}\, cd\wedge$$

$$\frac{\Gamma; \mathcal{D} \vdash D_1 \text{ compl} \qquad \Gamma; \mathcal{D} \vdash D_2 \text{ compl}}{\Gamma; \mathcal{D} \vdash D_1 \vee D_2 \text{ compl}}\, cd\vee$$

Figure 1.7: Complementable clause and goal: $\Gamma; \mathcal{D} \vdash D$ compl and $\Gamma; \mathcal{D} \vdash G$ compl

Example 1.20 *The clause encoding the introduction rule for implication in natural deduction from Example 5.1 is* not *complementable:*

$$impi \quad : \quad nd(A \ imp \ B) \leftarrow (nd(A) \rightarrow nd(B)).$$

On the other hand, the following is allowed by rule At:

$$\begin{array}{rcl} oplam^{\top} & : & \forall E{:}exp \rightarrow exp.\, open\ (lam\ E) \\ & & \quad \leftarrow (\forall x{:}exp.\, \top_{open} \rightarrow open\ (E\ x)). \end{array}$$

The restriction on goal and clause yields this revised grammar:

$$\begin{array}{lcll} \textit{Clauses} & D & ::= & \top \mid \bot \mid \forall(Q \leftarrow G) \mid D_1 \wedge D_2 \mid D_1 \vee D_2 \\ \textit{Goals} & G & ::= & Q \mid \top \mid \bot \mid \vec{M} \doteq \vec{N} \mid \vec{M} \not\doteq \vec{N} \mid \\ & & & G_1 \wedge G_2 \mid G_1 \vee G_2 \mid D \rightarrow G \mid \forall x{:}a.\, G \end{array}$$

We use $\forall(Q \leftarrow G)$ as a normal form for 'rules', where the quantifier bounds every free variable in $Q \leftarrow G$. This has the technical advantage to provide a handle for both the head of a clause and the set of all its free variables at the same time, which will be crucial to describe the complement algorithm. In particular, 'facts' are represented by $\forall(Q) \leftarrow \top$, although in examples we will omit to mention the body. We accordingly specialize the immediate implication and denial rules as follows:

$$\frac{\Gamma \vdash \sigma : FV(\vec{N}) \setminus \Gamma \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\vec{N} \doteq \vec{M} \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(q\ \vec{N} \leftarrow G) \gg q\ \vec{M}} \gg\rightarrow$$

$$\frac{\text{for all } \theta\ \Gamma \vdash \theta : FV(\vec{N}) \setminus \Gamma \qquad \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [\theta]\vec{N} \doteq \vec{M}}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \forall(q\ \vec{N} \leftarrow G) \gg q\ \vec{M}} \not\gg\rightarrow_1$$

$$\frac{\Gamma \vdash \sigma : FV(\vec{N}) \setminus \Gamma \qquad \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [\sigma]\vec{N} \not\doteq \vec{M} \qquad \not\vdash_{\mathcal{P}} [\sigma]G}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \forall(q\ \vec{N} \leftarrow G) \gg q\ \vec{M}} \not\gg\rightarrow_2$$

We remark that the 'guards' in the immediate denial rules are constructed so as to make the positive and negative judgment mutually independent. Thus $\not\gg \rightarrow_1$ says that a clause $\forall(q\ \vec{N} \leftarrow G)$ denies a goal $q\ \vec{M}$ if for all well-typed substitutions θ $[\theta]\vec{N}$ and $\vec{M}$ clashes. Rule $\not\gg \rightarrow_2$ instead attributes denial with the same conclusion due to 'failure' in the body.

Clause complementation is takes *definition-wise*. We cluster a program in a conjunction of possibly mutually recursive predicate definitions; We accomplish this with the following:

Definition 1.21 ($\Gamma \vdash$ def(q, D))

$$\begin{array}{rcll} \Gamma \vdash \mathit{def}(q, \top) & = & \top_q & \\ \Gamma \vdash \mathit{def}(q, \bot) & = & \bot_q & \\ \Gamma \vdash \mathit{def}(q, \forall(Q \leftarrow G)) & = & \forall(Q \leftarrow G) & \textit{if } Q \equiv q\ \vec{M} \\ \Gamma \vdash \mathit{def}(q, \forall(Q \leftarrow G)) & = & \top_q & \textit{otherwise} \\ \Gamma \vdash \mathit{def}(q, D_1 \wedge D_2) & = & \Gamma \vdash \mathit{def}(q, D_1) \wedge \Gamma \vdash \mathit{def}(q, D_2) & \\ \Gamma \vdash \mathit{def}(q, D_1 \vee D_2) & = & \Gamma \vdash \mathit{def}(q, D_1) \vee \Gamma \vdash \mathit{def}(q, D_2) & \end{array}$$

If $D = \mathcal{P}$, i.e. the underlying program (seen as a conjunction of clauses), we call $\cdot \vdash \text{def}(q, \mathcal{P})$ the *static* definition of q and avoid mentioning the parameter context when it can be inferred from the context. If $\Gamma; \mathcal{D}$ is a conjunction of a run-time context, we call the latter the *dynamic* definition of q. As a special case, an undefined predicate q, i.e. a predicate which occurs in a body of a clause but not as a head is represented by the empty conjunction $\top_q$. Conversely $\bot_q$ denotes the universal definition for q. We agree to consider $\top_q$ ($\bot_q$ resp.) as the zero (one) element for conjunction and disjunction and implicitly apply the appropriate absorption operations; for example:

$$\top_q \wedge \forall(q\ \vec{M} \leftarrow G) = \forall(q\ \vec{M} \leftarrow G)$$

. We overload the notation $D \sqsubseteq \mathcal{D}$ in the case $D \sqsubseteq \Gamma \vdash \mathrm{def}(q, \mathcal{P} \wedge \mathcal{D})$ to indicate the relation that satisfies the following rules:

$$\frac{D \in \{\top, \bot, \forall(Q \leftarrow G)\}}{D \sqsubseteq D} \qquad \frac{D_1 \sqsubseteq \mathcal{D} \qquad D_2 \sqsubseteq \mathcal{D}}{D_1 \wedge D_2 \sqsubseteq \mathcal{D}} \qquad \frac{D_1 \sqsubseteq \mathcal{D} \qquad D_2 \sqsubseteq \mathcal{D}}{D_1 \vee D_2 \sqsubseteq \mathcal{D}}$$

The immediate implication and denial judgments is specialized as obvious:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ Clause D from $\Gamma \vdash \mathrm{def}(q, \mathcal{P} \wedge \mathcal{D})$ immediately entails atom Q.
$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \gg Q$ Clause D from $\Gamma \vdash \mathrm{def}(q, \mathcal{P} \wedge \mathcal{D})$ immediately denies atom Q.

In particular, the 'At' rules are:

$$\frac{D \sqsubseteq \mathrm{def}(Q, \mathcal{P} \wedge \mathcal{D}) \qquad \lceil\Gamma\rceil; \lceil\mathcal{D}\rceil \vdash_{\mathcal{P}} D \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} Q} \vdash \mathrm{At}$$

$$\frac{D \sqsubseteq \mathrm{def}(Q, \mathcal{P} \wedge \mathcal{D}) \qquad \lceil\Gamma\rceil; \lceil\mathcal{D}\rceil \not\vdash_{\mathcal{P}} D \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} Q} \not\vdash \mathrm{At}$$

Accordingly, we also specialized the rule for schema satisfaction: we use ρ to denote a *global* substitution, e.g. such that $\mathrm{dom}(\rho)$ is a set of new global parameters.

$$\frac{\cdot; \top \setminus [\rho]G < \mathcal{S}}{\models_{\mathcal{S}} \forall(Q \leftarrow G) \gg q\ \vec{M}} \models_{\mathcal{S}} \to^{\rho}$$

It is clear that all the above rules are derived rules of inference, as can be proven by a straightforward induction on the number of free variables in $\vec{N}$.

1.4.1 Normalization of Input Variables

We present in Figure 6.8 the rules for normalization of input variables. Since we are currently working with the restriction to ground goals, those are all the variables which are universally quantified in the head of a clause. When they occur positively in dynamic assumptions, they will carry some instantiation. It helps to simplify the presentation of the clause complementation algorithm if we forbid this kind of occurrence. The idea it to replace every positive occurrence of an input variable in an assumption with a new (local) universal variable, which is then constrained to be equal to the input ones.

Example 1.22 *Consider the typing clause for lambda terms:*

$$\begin{array}{lll} oflam & : & \forall E : exp \to exp.\, \forall T_1, T_2 : tp. \\ & & \quad of\ (lam\ E)\ (arrow\ T_1\ T_2) \\ & & \qquad \leftarrow (\forall x : exp.\, of\ x\ T_1 \to of\ (E\ x)\ T_2). \end{array}$$

As T_1 is an input term occurring positively in the assumption 'of x T_1', normalization of input variables inserts a new variable T and constrains it to be equal to T_1:

$$\begin{array}{lll} oflam' & : & \forall E : exp \to exp.\, \forall T_1, T_2 : tp. \\ & & \quad of\ (lam\ E)\ (arrow\ T_1\ T_2) \\ & & \qquad\qquad (\forall T : tp.\, of\ x\ T \leftarrow T \doteq T_1) \to \\ & & \qquad\qquad\quad of\ (E\ x)\ T_2). \end{array}$$

This procedure is realized by the judgments:

- $D \overset{D}{\mapsto} D'$: clause normalization.

- $\Gamma \vdash_\Phi G \overset{G}{\mapsto} G'$: goal normalization.
- $\Phi; \Gamma \vdash D \mapsto D'$: assumption normalization w.r.t. global parameters in Φ.
- $\Lambda; \Phi; \Gamma \vdash M \mapsto M', E$: term M normalizes to term M' returning a conjunction of equations E w.r.t. bound (Γ), global (Φ) and existential (Λ) variables.

The first three judgments simply recur on the structure of clauses, goal and assumptions, keeping track of bound, global and existential variables in assumptions, until we normalize an assumption $\forall(Q \leftarrow G)$: here we descend in every term by introducing new local existential variable and binding them to the global one by an equation E. For the sake of conciseness, we do this for the simply-typed fragment; the generalization to the strict one is immediate. In rule $\overset{D}{\mapsto}\leftarrow^{\rho}$ we introduce in the global context all the global parameters in the domain of ρ with the appropriate typing.

It is clear, although very tedious, to verify that this transformation preserves provability and denial,

Lemma 1.23 *Assume $\Lambda; \Phi; \Gamma \vdash M \mapsto M', E$; for every ground N, every $\Gamma; \mathcal{D}$, $\Gamma; \mathcal{D} \vdash M \doteq N$ iff $\Gamma; \mathcal{D} \vdash M' \doteq N \wedge E$.*

Proof: By a straightforward induction on the structure of $\Lambda; \Phi; \Gamma \vdash M \mapsto M', E$. □

Lemma 1.24 *Let $\Gamma \vdash \theta : \Phi$ and $\Phi; \Gamma \vdash D \mapsto D'$. For every $\mathcal{D}$, it holds $\Gamma; [\theta]\mathcal{D} \vdash [\theta]D \gg Q$ iff $\Gamma; [\theta]\mathcal{D} \vdash [\theta]D' \gg Q$.*

Proof: By a straightforward induction on the structure of $\Phi; \Gamma \vdash D \mapsto D'$, using Lemma 6.23. □

Theorem 1.25 *Let $\Gamma \vdash \theta : \Phi$ $D \overset{D}{\mapsto} D'$ and $\Gamma \vdash_\Phi G \overset{G}{\mapsto} G'$:*

1. *$\Gamma; [\theta]\mathcal{D} \vdash [\theta]D \gg Q$ iff $\Gamma; [\theta]\mathcal{D} \vdash [\theta]D' \gg Q$.*
2. *$\Gamma; [\theta]\mathcal{D} \vdash [\theta]G$ iff $\Gamma; [\theta]\mathcal{D} \vdash [\theta]G'$.*

Proof: By a straightforward mutual induction on the structure of $D \overset{D}{\mapsto} D'$ and $\Gamma \vdash_\Phi G \overset{G}{\mapsto} G'$, using Lemma 6.24. □

1.5 The Clause Complement Algorithm

We now introduce in Figure 6.9 and Figure 6.10 the rules for static and dynamic clause complementation. Consider a rule $\forall(q\ \vec{M} \leftarrow G)$; its complement must contain a 'factual' part motivating failure due to clash with the head; the remainder $\mathrm{Not_G}(G)$ expresses failure in the body, if any. Clause complementation must discriminate whether this rule belongs to the static or dynamic definition of a predicate. In the first case all the relevant information is already present in the head of the clause and we can use the term complementation algorithm described in Chapter 4. This is accomplished by the rule $\mathrm{Not_D} \rightarrow$, where a set of negative facts is built via term complementation $\mathrm{Not}(\vec{M})$ applied in the empty context to the (vector of) terms in the clause head; namely $\bigwedge_{\vec{N} \in \cdot \vdash \mathrm{Not}(\vec{M})} \forall(\neg(q\ \vec{N}) \leftarrow \top)$; moreover the negative counterpart of the source clause is obtained via complementation of the body. The Partition Lemma (Corollary 4.21) guarantees the soundness and completeness of this case.

Assumption complementation is realized by the judgment $\Gamma \vdash \mathrm{Not}_\alpha(D)$, which can be seen as a type directed *parameter-conscious* version of clause complementation. In a first approximation, we can think of complementation of assumptions, which are by definition parametric in some x, as static clause complementation w.r.t. x. Informally, for an atomic assumption, say $q\ M_1 \ldots M_{i-1}\ e_x\ M_{i+1} \ldots M_n$, its complement can be taken as $\mathrm{Not_D}(q_{e_x}\ M_1 \ldots M_{i-1}\ M_{i+1} \ldots M_n)$. This is accomplished in two phases: first, e_x is propagated in every position $1 \leq i \leq n$ holding a rigid term of compatible type, but different from e_x itself. This alone builds an element of the complement set. Secondly, the idea is to can apply term complementation 'around' e_x. In particular, if M_i is a variable, by normalization of input variables, it must be a local one and term complement does not contribute anything as expected. If M_i is a parameter expression or a compound term,

$$\frac{D_1 \overset{D}{\mapsto} D_1' \qquad D_2 \overset{D}{\mapsto} D_2'}{D_1 \wedge D_2 \overset{D}{\mapsto} D_1' \wedge D_2'} \overset{D}{\mapsto} \wedge \qquad \frac{D_1 \overset{D}{\mapsto} D_1' \qquad D_2 \overset{D}{\mapsto} D_2'}{D_1 \vee D_2 \overset{D}{\mapsto} D_1' \vee D_2'} \overset{D}{\mapsto} \vee$$

$$\frac{X \in \{\top, \bot\}}{X \overset{D}{\mapsto} X} \overset{D}{\mapsto} X \qquad \frac{\cdot \vdash_{\mathrm{dom}(\rho)} [\rho]G \overset{G}{\mapsto} [\rho]G'}{\forall(Q \leftarrow G) \overset{D}{\mapsto} \forall(Q \leftarrow G')} \overset{D}{\mapsto}\leftarrow^{\rho}$$

$$\frac{X \in \{\top, \bot, Q\} \text{ or (dis)eq}}{\Gamma \vdash_\Phi X \overset{G}{\mapsto} X} \overset{G}{\mapsto} X \qquad \frac{\Gamma, y{:}A); \mathcal{D} \vdash_\Phi [y/x]G \overset{G}{\mapsto} [y/x]G'}{\Gamma \vdash_\Phi \forall x : a.\, G \overset{G}{\mapsto} \forall x : a.\, G'} \overset{G}{\mapsto} \forall^y$$

$$\frac{\Phi; \Gamma \vdash D \mapsto D' \qquad \Gamma; \mathcal{D} \wedge D \vdash_\Phi G \overset{G}{\mapsto} G'}{\Gamma \vdash_\Phi (D \to G) \overset{G}{\mapsto} (D' \to G')} \overset{G}{\mapsto}\to$$

$$\frac{\Gamma \vdash_\Phi G_1 \overset{G}{\mapsto} G_1' \qquad \Gamma \vdash_\Phi G_2 \overset{G}{\mapsto} G_2'}{\Gamma \vdash_\Phi G_1 \wedge G_2 \overset{G}{\mapsto} G_1' \wedge G_2'} \overset{G}{\mapsto} \wedge \qquad \frac{\Gamma \vdash_\Phi G_1 \overset{G}{\mapsto} G_1' \qquad \Gamma \vdash_\Phi G_2 \overset{G}{\mapsto} G_2'}{\Gamma \vdash_\Phi G_1 \vee G_2 \overset{G}{\mapsto} G_1' \vee G_2'} \overset{G}{\mapsto} \vee$$

$$\frac{}{\Lambda; \Phi; (\Gamma, x{:}A) \vdash x \mapsto x, \top}\, bv \qquad \frac{Z \text{ new}}{\Lambda; (\Phi, u{:}A); \Gamma \vdash u \mapsto Z, Z \doteq u}\, iv \qquad \frac{}{(\Lambda, x{:}A); \Phi; \Gamma \vdash x \mapsto x, \top}\, lv$$

$$\frac{\Lambda; \Phi; \Gamma \vdash M_1 \mapsto M_1', E_1 \qquad \Lambda; \Phi; \Gamma \vdash M_2 \mapsto M_2', E_2}{\Lambda; \Phi; \Gamma \vdash M_1\ M_2 \mapsto M_1'\ M_2', E_1 \wedge E_2}\, App \qquad \frac{\Lambda; \Phi; (\Gamma, x{:}A) \vdash M \mapsto M', E}{\Lambda; \Phi; \Gamma \vdash \lambda x : A.\, M \mapsto \lambda x : A.\, M', E}\, \lambda$$

$$\frac{\Lambda = FV(\overline{M_n}) \setminus (\Gamma \cup \Phi) \qquad \Lambda; \Phi; \Gamma \vdash M_1 \mapsto M_1', E_1 \ldots \Lambda; \Phi; \Gamma \vdash M_n \mapsto M_n', E_n}{\Phi; \Gamma \vdash \forall(q\ \overline{M_n} \leftarrow G) \mapsto \forall(q\ \overline{M_n'} \leftarrow E_1 \wedge \ldots \wedge E_n \wedge G)} \mapsto\to$$

$$\frac{X \in \{\top, \bot\}}{\Phi; \Gamma \vdash X \mapsto X} \mapsto X$$

$$\frac{\Phi; \Gamma \vdash D_1 \mapsto D_1' \qquad \Phi; \Gamma \vdash D_2 \mapsto D_2'}{\Phi; \Gamma \vdash D_1 \wedge D_2 \mapsto D_1' \wedge D_2'} \mapsto \wedge \qquad \frac{\Phi; \Gamma \vdash D_1 \mapsto D_1' \qquad \Phi; \Gamma \vdash D_2 \mapsto D_2'}{\Phi; \Gamma \vdash D_1 \vee D_2 \mapsto D_1' \vee D_2'} \mapsto \vee$$

Figure 1.8: Clause, goal, assumption and term normalization

$$\frac{}{\mathrm{Not_D}(\top) = \bot}\,\mathrm{Not_D}\top \qquad \frac{}{\mathrm{Not_D}(\bot) = \top}\,\mathrm{Not_D}\bot$$

$$\frac{\cdot \vdash \mathrm{Not_G}(G) = G'}{\mathrm{Not_D}(\forall(q\ \vec{M} \leftarrow G)) = \displaystyle\bigwedge_{\vec{N} \in \cdot \vdash \mathrm{Not}(\vec{M})} \forall(\neg(q\ \vec{N}) \leftarrow \top) \wedge \forall(\neg q\ \vec{M} \leftarrow G')}\,\mathrm{Not_D} \leftarrow$$

$$\frac{\mathrm{Not_D}(D_1) = D_1' \qquad \mathrm{Not_D}(D_2) = D_2'}{\mathrm{Not_D}(D_1 \wedge D_2) = D_1' \vee D_2'}\,\mathrm{Not}\wedge$$

$$\frac{\mathrm{Not_D}(D_1) = D_1' \qquad \mathrm{Not_D}(D_2) = D_2'}{\mathrm{Not_D}(D_1 \vee D_2) = D_1' \wedge D_2'}\,\mathrm{Not}\vee$$

Figure 1.9: Clause Complementation: $\mathrm{Not_D}(D) = D'$

we simply take the term complement of the former term, with the notable difference of passing the current parameter context to term complement.

However, not all parameters are born alike; in many situations, a parametric judgment is used simply to descend into a scoping construct, while the parameter itself does not play a role w.r.t. provability and denial. Consider, for example, the following program that checks whether a universal formula is a D-clause:

$$\begin{array}{rcl} form, term & : & type. \\ all & : & (term \to form) \to form). \\ isd & : & form \to \mathbf{o} \\ isdall & : & \forall D{:}term \to form.\ isd\ (all\ D) \\ & & \quad \leftarrow \forall x{:}term.\ isd\ (D\ x). \end{array}$$

This phenomenon is known in the literature under the name of *subordination* and has been extensively studies in the dependent typed context [?]. In the simply typed setting, this relation collapses to merely checking whether the type of the parameters is equal to the target type of some argument of the predicate. More formally, we say that $x{:}A$ is *relevant to* Q if $head(Q) \equiv q$, $\Sigma(q) = A_1, \ldots, A_n \to \mathbf{o}$ and for some $1 \leq i \leq n$ it holds that $\mathrm{target}(A_i) \equiv A$; we denote this with xR^iq. In the above example, as $term \not\equiv form$, then $x : term$ is *not* relevant to $isd : form \to \mathbf{o}$. Wrt. clause complementation, if the parameter x is *not* relevant to q, we do not need to build complementary facts out of it, although they would not impact soundness, i.e. Exclusivity (Theorem 6.34).

We concentrate on rules $\mathrm{Not}_\alpha\top$ and $\mathrm{Not}_\alpha \to$. The notation $[e_x/Z_i]\overline{Z_n}$ is an abbreviation for $Z_1 \ldots Z_{i-1}\ e_x\ Z_{i+1} \ldots Z_n$, where the Z's are fresh logic variables; similarly for $[e_x/Z_i, N/Z_j]\overline{Z_n}$. The main loop goes as follows:

- Choose a parameter $x{:}a \in \Gamma$.
- Propagate e_x in every 'odd' position.
- Locate a type A_i such that x is relevant to Q at i:
  - Complement D w.r.t. x and i.
  - Repeat for every relevant position i.
- Repeat for every x.

$$\frac{}{\Gamma \vdash \mathrm{Not}_\alpha(\bot) = \top}\,\mathrm{Not}_\alpha\bot$$

$$\frac{\Sigma(q) = A_1 \to \cdots \to A_n \to \mathbf{o}}{\Gamma \vdash \mathrm{Not}_\alpha(\top_q) = \bigwedge_{x \in dom(\Gamma)} (\bigwedge_{1 \leq i \leq n, xR^i q} \mathrm{Not}^C_x(\top_q))}\,\mathrm{Not}_\alpha\top$$

$$\frac{\Gamma \vdash \mathrm{Not}_{\mathrm{G}}(G) = G'}{\Gamma \vdash \mathrm{Not}_\alpha(\forall(q\ \overline{M_n} \leftarrow G)) = (\bigwedge_{x \in dom(\Gamma)} \mathrm{Not}^C_x(q\ \overline{M_n}) \wedge (\bigwedge_{1 \leq i \leq n, xR^i q} \Gamma \vdash \mathrm{Not}^i_x(q\ \overline{M_n}))) \wedge \forall(\neg q\ \overline{M_n}) \leftarrow G')}\,\mathrm{Not}_\alpha \leftarrow$$

$$\frac{\Gamma \vdash \mathrm{Not}_\alpha(D_1) = D_1' \qquad \Gamma \vdash \mathrm{Not}_\alpha(D_2) = D_2'}{\Gamma \vdash \mathrm{Not}_\alpha(D_1 \wedge D_2) = D_1' \vee D_2'}\,\mathrm{Not}\wedge$$

$$\frac{\Gamma \vdash \mathrm{Not}_\alpha(D_1) = D_1' \qquad \Gamma \vdash \mathrm{Not}_\alpha(D_2) = D_2'}{\Gamma \vdash \mathrm{Not}_\alpha(D_1 \vee D_2) = D_1' \wedge D_2'}\,\mathrm{Not}\vee$$

$$\frac{M_i : A_i \text{ rigid}, \cdot \vdash sh(x, A_i) = e_x, M_i \not\equiv e_x}{\mathrm{Not}^C_x(q\ \overline{M_n}) = \bigwedge_{1 \leq i \leq n} \forall Z_1 : A_1. \ldots \forall Z_n : A_n. \neg q\ [e_x/Z_i]\overline{Z_n}}\,\mathrm{Not}^C_x$$

$$\frac{M_i \equiv e_x}{\Gamma \vdash \mathrm{Not}^i_x(q\ \overline{M_n}) = \bigwedge_{1 \leq j \leq n}^{j \neq i} (\bigwedge_{N \in (\Gamma \vdash \mathrm{Not}(M_j))} (\forall Z_1 : A_1. \ldots \forall Z_n : A_n. \neg q\ [e_x/Z_i, N/Z_j]\overline{Z_n}))}\,\mathrm{Not}^i_x$$

$$\frac{}{\Gamma \vdash sh(x, a) = x}\,e_x\mathrm{At} \qquad \frac{\Gamma, y{:}A \vdash sh(x, B) = e_x}{\Gamma \vdash sh(x, A \to B) = \lambda y{:}A.\, e_x}\,e_x \to^y$$

Figure 1.10: Assumption complementation: $\Gamma \vdash \mathrm{Not}_\alpha(D) = D'$

Rule $\mathrm{Not}_\alpha \rightarrow$ is the most complicated one: fixed a parameter x, there are two ways in which an atomic assumption $q\ M_1 \ldots M_{i-1}\ e_x\ M_{i+1} \ldots M_n$, needs to be complemented. First, any atom with same head with a term $N_j \equiv e_x$ for $i \neq j$ is in the complement of the former. Moreover, since they do differ in one coordinate, we may as well leave open every other term. This is encoded in rule Not_x^C. Of course, we have to make sure that N_j is not flex and that the type is appropriate. Since x is passed as a parameter, the $\Gamma \vdash sh(x, A)$ judgment builds a shallow parameter expression as required by the type of the position in q where it ought to occur.

Secondly, we have to take into account the case $M_i \equiv e_x$: here we can build a set of complementary facts by pivoting on e_x and making another position different. Luckily, we can achieve this via a call to term complementation and build a set of complementary facts similarly to clause complementation. The difference is that we pass to term complementation, as a context, the set of parametric bound variables. Finally, both processes are repeated for every M_i.

Notice the different treatment of the trivial clause $\top$ by rules $\mathrm{Not_D}\top$ and $\mathrm{Not}_\alpha\top$: if no parameter has been assumed, then $\top$ truly stands for the empty predicate definition and its complement is the universal definition $\bot$. If, on the other side Γ is *not* empty, it means that $\top_q$ has been introduced during the $\top$-normalization preprocessing phase and has been localized to the predicate q. Here we need to construct a new negative assumption w.r.t. q, x, i in case $\top_q$ is the only dynamic definition of q. As $\top_q$ carries no information at all concerning q, the most general negative assumption is added. This is accomplished again by rule Not_x^C, where we make the convention to view $\top_q$ as a degenerate case of $q\ \overline{M_n}$ where the sequence $\overline{M_n}$ is empty (and thus the condition trivially satisfied).

The remaining (common) rules for static and dynamic clause complementation simply recur on the program respecting the duality of conjunction and disjunction w.r.t. negation. This a somewhat delicate point and therefore we discuss it in some details. Intuitively, negative clauses stemming from the complement of a clause in the definition of a predicate need to be considered simultaneously. In fact, if an goal is unprovable from its definition, then its negation must be provable from the complement of each clause of its definition; symmetrically if it is provable then its negation must be unprovable from the complement of at least one. What we have described coincides the operational semantics of an operator, which works exactly as disjunction. This is arguably at odd with the commonly held goal-oriented interpretation of the sequent calculus as uniform proofs, since case analysis makes the latter incomplete w.r.t. minimal logic (but see [?] for ways to incorporate the former in the framework of uniform proofs). In Section ?? we will show how to 'compile away' all occurrences of $\vee$ in clauses. This is an higher-order equivalent of the intersection operator '@' described in [?] and will restore completeness of uniform proofs

Goal complementation, that is the judgment $\mathrm{Not_G}(G)$ depicted in Figure 6.11 is straightforward, since it only brings the body into a normalized format; namely, in what we may call *parametric negation normal form*, to stress the distinction from classical negation normal form or even from negation normal form in constructive logics as extension of intuitionism with strong negation [?]. This re-iterates the problem with strong negation we have hinted in Section 5.3; negation is pushed inward but jumps over parametric-hypothetical judgments to respect the operational interpretation of unprovability.

As a final remark, we note again that we must take the complementation of a program, seen as a conjunction, *predicate definition-wise* rather that *clause-wise*. In fact, it would be incorrect to simply negate a program as it would introduce disjunctions rather than conjunction among predicate definitions. The same remark applies to the 'dynamic' program $\mathcal{D}$. Formally, given a fixed signature $\Sigma_\mathcal{P}$ and a program $\mathcal{P}$:

$$\mathrm{Not_D}(\mathcal{P}) \quad = \quad \bigwedge_{q \in \Sigma_\mathcal{P}} \mathrm{Not_D}(\mathrm{def}(q, \mathcal{P}))$$

Similarly for a (possibly run-time) context$\Gamma; \mathcal{D}$:

$$\Gamma \vdash \mathrm{Not}_\alpha(\mathcal{D}) \quad = \quad \bigwedge_{q \in \Sigma_\mathcal{P}} \Gamma \vdash \mathrm{Not}_\alpha(\mathrm{def}(q, \mathcal{D}))$$

Note that if $\Gamma; \mathcal{D}$ is a run-time context, it consists of a conjunction of blocks, i.e. it has the form $\Gamma'', \lceil\Gamma'\rceil; \mathcal{D}'' \wedge \lceil\mathcal{D}'\rceil$; in this case the definition of q consists of the conjunction of the definition for *every*

$$\frac{}{\Gamma \vdash \mathrm{Not_G}(\top) = \bot}\,\mathrm{Not_G}\top \qquad \frac{}{\Gamma \vdash \mathrm{Not_G}(\bot) = \top}\,\mathrm{Not_G}\bot$$

$$\frac{}{\Gamma \vdash \mathrm{Not_G}(\vec{M} \doteq \vec{N}) = (\vec{M} \not\doteq \vec{N})}\,\mathrm{Not}\doteq \qquad \frac{}{\Gamma \vdash \mathrm{Not_G}(\vec{M} \not\doteq \vec{N}) = (\vec{M} \doteq \vec{N})}\,\mathrm{Not}\not\doteq$$

$$\frac{}{\Gamma \vdash \mathrm{Not_G}(Q) = \neg Q}\,\mathrm{Not_G}At$$

$$\frac{\Gamma, y{:}a \vdash \mathrm{Not_G}([y/x]G) = [y/x]G'}{\Gamma \vdash \mathrm{Not_G}(\forall x : a.\, G) = \forall x : a.\, G'}\,\mathrm{Not_G}\forall^y$$

$$\frac{\Gamma \vdash \mathrm{Not_G}(G_1) = G_1' \qquad \Gamma \vdash \mathrm{Not_G}(G_2) = G_2'}{\Gamma \vdash \mathrm{Not_G}(G_1 \wedge G_2) = G_1' \vee G_2'}\,\mathrm{Not}\wedge$$

$$\frac{\Gamma \vdash \mathrm{Not_G}(G_1) = G_1' \qquad \Gamma \vdash \mathrm{Not_G}(G_2) = G_2'}{\Gamma \vdash \mathrm{Not_G}(G_1 \vee G_2) = G_1' \wedge G_2'}\,\mathrm{Not}\vee$$

$$\frac{\Gamma \vdash \mathrm{Not_G}(G) = G'}{\Gamma \vdash \mathrm{Not_G}(D \to G) = D \to G'}\,\mathrm{Not_G}\to$$

Figure 1.11: Goal complementation: $\Gamma \vdash \mathrm{Not_G}(G) = G'$

block, namely: $\Gamma' \vdash \mathrm{def}(q, \mathcal{D}') \wedge \Gamma'' \vdash \mathrm{def}(q, \mathcal{D}'')$. We abbreviate $\mathrm{Not_D}(\mathrm{def}(q))$ in $\mathrm{def}(\neg q)$. If $\mathcal{P}^+$ is the source program, we use $\mathcal{P}^-$ for $\mathrm{Not_D}(\mathcal{P})$.

Finally, we provide an example, which, even though is somewhat a special case of the general procedure, assumption complementation being trivial, it helps to clarify the rules for clause and goal complementation in isolation. We refer to the next Section (6.6) for more complex examples. We use $\forall F_1, F_2 : A.\, X$ as an abbreviation of $\forall F_1 : A.\, \forall F_2 : A.\, X$.

Example 1.26 *A λ-expression is* closed *if it has no occurrence of free variables; let $z : exp$ be a constant:*

$$\begin{array}{rcl} cloz & : & closed\ z. \\ clolam & : & \forall E : exp.\, closed\ (lam\ E) \leftarrow (\forall x : exp.\ (closed\ x \to closed\ (E\ x).) \\ cloapp & : & \forall E_1 : exp.\, \forall E_2 : exp.\, closed\ (app\ E_1\ E_2) \leftarrow closed\ E_1 \wedge closed\ E_2. \end{array}$$

Now, $def(closed) = cloz \wedge clolam \wedge cloapp$. Note that $x{:}exp \vdash \mathrm{Not}_\alpha(closed\ x) = \top$. Therefore:

$$def(\neg closed) = \mathrm{Not_D}(cloz) \vee \mathrm{Not_D}(clolam) \vee \mathrm{Not_D}(cloapp)$$

$$\begin{array}{rcl} \mathrm{Not_D}(cloz) & = & \displaystyle\bigwedge_{N \in \mathrm{Not}(z)} \neg\forall(closed\ N) \\ & = & (\forall F : exp.\, \neg closed\ (lam\ F)) \wedge \forall F_1, F_2 : exp.\, \neg closed\ (app\ F_1\ F_2). \\ \mathrm{Not_D}(clolam) & = & \displaystyle\bigwedge_{N \in \mathrm{Not}(lam\ E)} \forall(\neg closed\ N) \wedge \\ & & \forall E : exp.\, \neg closed\ (lam\ E) \leftarrow \mathrm{Not_G}(\forall x : exp.\ (closed\ x \to closed\ (E\ x)) \\ & = & \neg closed\ z \wedge (\forall F_1, F_2 : exp.\, \neg closed\ (app\ F_1\ F_2)) \wedge \\ & & \forall E : exp.\, \neg closed\ (lam\ E) \leftarrow (\forall x : exp.\ (closed\ x \to \neg closed\ (E\ x)). \end{array}$$

$$\frac{X \in \{\top, \bot\}}{aug_{\mathrm{D}}(X) = X}\, aug_{\mathrm{D}}X$$

$$\frac{aug_{\mathrm{D}}(D_1) = D_1^a \qquad aug_{\mathrm{D}}(D_2) = D_2^a}{aug_{\mathrm{D}}(D_1 \wedge D_2) = D_1^a \wedge D_2^a}\, aug_{\mathrm{D}}\wedge$$

$$\frac{aug_{\mathrm{D}}(D_1) = D_1^a \qquad aug_{\mathrm{D}}(D_2) = D_2^a}{aug_{\mathrm{D}}(D_1 \vee D_2) = D_1^a \vee D_2^a}\, aug_{\mathrm{D}}\vee$$

$$\frac{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}([\rho]G) = [\rho]G^a}{aug_{\mathrm{D}}(\forall(Q \leftarrow G)) = \forall(Q \leftarrow G^a)}\, aug_{\mathrm{D}} \leftarrow^{\rho}$$

Figure 1.12: Clause augmentation: $aug_{\mathrm{D}}(D) = D^a$

$$\begin{array}{rcl}
\mathrm{Not_D}(cloapp) & = & \bigwedge_{N \in \mathrm{Not}(app\ E_1\ E_2))} \neg\forall(closed\ N) \wedge \\
& & \forall E_1, E_2 : exp.\, \neg closed\ (app\ E_1\ E_2) \leftarrow \mathrm{Not_G}(closed\ E_1 \wedge closed\ E_2) \\
& = & \neg closed\ z \wedge (\forall F : exp.\, \neg closed(lam\ F)) \wedge \\
& & \forall E_1, E_2 : exp.\, \neg closed\ (app\ E_1\ E_2) \leftarrow \neg closed\ E_1 \vee \neg closed\ E_2.
\end{array}$$

If you want to see the definition simplified, please skip to Example ?? and the final result in Example ??.

1.6 Augmentation

Now that we have discussed how to perform clause, assumption and goal complementation, we synchronize it together in a phase we call *augmentation*, which simply inserts the correct assumption complementation in a goal and in turn in a clause. We give one judgment to augment a program, $aug_{\mathrm{D}}(D)$, depicted in 6.12. which merely recurs on the structure of clauses until it calls goal augmentation, $\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(G)$ (Figure 6.12). The latter traverses a goal collecting parameters in Γ and assumptions in $\mathcal{D}$. When it reaches an atom, either it stops, as no parameter has been introduced, or it passes $\Gamma, \mathcal{D}$ to assumption complementation.

Some examples will make the whole process clear; consider the `copy` program on λ-terms:

$$\begin{array}{rcl}
cpapp & : & \forall E_1, E_2, F_1, F_2 : exp. \\
& & copy\ (app\ E_1 E_2)\ (app\ F_1 F_2) \\
& & \quad \leftarrow copy\ E_1\ F_1 \\
& & \quad \leftarrow copy\ E_2\ F_2. \\
cplam & : & \forall E : exp \rightarrow exp.\, \forall F : exp \rightarrow exp. \\
& & copy\ (lam\ E)\ (lam\ F) \\
& & \quad \leftarrow (\forall x : exp.\, copy\ x\ x \\
& & \qquad \rightarrow copy\ (E\ x)\ (F\ x)).
\end{array}$$

The augmentation judgment $aug_{\mathrm{D}}(cplam)$ calls $aug_{\mathrm{G}}(\forall x : exp.\, copy\ x\ x \rightarrow copy\ (E\ x)\ (F\ x))$, which collects the context $x{:}exp; copy\ x\ x$ and calls $x{:}exp \vdash \mathrm{Not}_{\alpha}(copy\ x\ x)$. We start by noting that $\mathrm{Not}_x^C(copy\ x\ x) = \top$, since the conditions on rule Not_x^C are not satisfied. Then, note that $x{:}exp \vdash \mathrm{Not}(x) = \{lam\ F', app\ F_1\ F_2\}$. This yields:

$$x{:}exp \vdash \mathrm{Not}_1^x(copy\ x\ x) = (\forall F', F_1, F_2 : exp.\, \neg copy\ x\ (lam\ F') \wedge \neg copy\ x\ (app\ F_1\ F_2))$$

$$\frac{X \in \{\top, \bot\} \text{ or (dis)eq}}{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(X) = X}\, aug_{\mathrm{G}}X$$

$$\frac{}{\cdot; \top \vdash aug_{\mathrm{G}}(Q) = Q}\, aug_{\mathrm{G}}\mathrm{At}^{\top}$$

$$\frac{\Gamma \vdash \mathrm{Not}_{\alpha}(\mathcal{D}) = \mathcal{D}_{\neg}}{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(Q) = \mathcal{D}_{\neg} \to Q}\, aug_{\mathrm{G}}At$$

$$\frac{\Gamma; (\mathcal{D} \wedge D) \vdash aug_{\mathrm{G}}(G) = G^{a}}{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(D \to G) = D \to G^{a}}\, aug_{\mathrm{G}} \to$$

$$\frac{(\Gamma, y{:}a); \mathcal{D} \vdash aug_{\mathrm{G}}([y/x]G) = [y/x]G^{a}}{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(\forall x : a.\, G) = \forall x : a.\, G^{a}}\, aug_{\mathrm{G}}\forall^{y}$$

$$\frac{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(G_1) = G_1^{a} \qquad \Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(G_2) = G_2^{a}}{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(G_1 \wedge G_2) = G_1^{a} \wedge G_2^{a}}\, aug_{\mathrm{G}}\wedge$$

$$\frac{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(G_1) = G_1^{a} \qquad \Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(G_2) = G_2^{a}}{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(G_1 \vee G_2) = G_1^{a} \vee G_2^{a}}\, aug_{\mathrm{G}}\vee$$

Figure 1.13: Goal augmentation: $\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(G) = G^{a}$

Symmetrically:

$$x{:}exp \vdash \mathrm{Not}_2^{x}(copy\ x\ x) = \forall F'', F_3, F_4 : exp.\, \neg copy\ (lam\ F'')\ x \wedge \neg copy\ (app\ F_3\ F_4)\ x)$$

This will yield the augmented clause:

$$\begin{array}{lll} aug_{\mathrm{D}}(cplam) & : & \forall E : exp \to exp.\, \forall F : exp \to exp. \\ & & copy\ (lam\ E)\ (lam\ F) \\ & & \quad \leftarrow (\forall x : exp. \\ & & \quad (\forall F', F'', F_1, F_2, F_3, F_4 : exp. \\ & & \qquad \neg copy\ x\ (lam\ F') \wedge \neg copy\ x\ (app\ F_1\ F_2) \wedge \\ & & \qquad \neg copy\ (lam\ F'')\ x \wedge \neg copy\ (app\ F_3\ F_4)\ x) \\ & & \qquad\quad \to copy\ x\ x \to copy\ (E\ x)\ (F\ x)). \end{array}$$

while of course, $aug_{\mathrm{D}}(cpapp) = cpapp$, as enforced by $aug_{\mathrm{G}}\mathrm{At}\top$. If we had a two-parameter version of *copy*:

$$\begin{array}{lll} cplam' & : & \forall E : exp \to exp.\, \forall F : exp \to exp. \\ & & copy'\ (lam\ E)\ (lam\ F) \\ & & \quad \leftarrow (\forall x : exp.\, \forall y : exp.\, (copy'\ x\ y) \\ & & \qquad \to copy'\ (E\ x)\ (F\ y)). \end{array}$$

Then we would get first:

$$\begin{array}{lll} \mathrm{Not}_x^{C}(copy'\ x\ y) & = & (\forall E'' : exp.\, \neg copy'\ E''\ x) \\ \mathrm{Not}_y^{C}(copy'\ x\ y) & = & (\forall F'' : exp.\, \neg copy'\ y\ F''). \end{array}$$

Secondly, by computing, respectively, for $\Gamma = x{:}exp, y{:}exp$, $\Gamma \vdash \mathrm{Not}(y)$ and $\Gamma \vdash \mathrm{Not}(x)$:

$$\begin{array}{rcl}
\Gamma \vdash \mathrm{Not}^x_1(copy'\ x\ y) & = & (\forall F_0, F_1, F_2 : exp.\, \neg copy'\ x\ (lam\ F_0) \wedge \neg copy'\ x\ (app\ F_1\ F_2) \wedge \neg copy' x\ x) \\
\Gamma \vdash \mathrm{Not}^y_2(copy'\ x\ y) & = & (\forall F'_0, F'_1, F'_2 : exp.\, \neg copy'\ (lam\ F'_0)\ y \wedge \neg copy'\ (app\ F'_1\ F'_2)\ y \wedge \neg copy'\ y\ y)
\end{array}$$

yielding the augmented clause:

$$\begin{array}{rcl}
aug_{\mathrm{D}}(cplam') & : & \forall E : exp \to exp.\, \forall F : exp \to exp. \\
& & copy'\ (lam\ E)\ (lam\ F) \\
& & \quad \leftarrow (\forall x : exp.\, \forall y : exp. \\
& & \qquad (\forall E' : exp.\, \neg copy'\ E'\ x) \wedge \\
& & \qquad (\forall F' : exp.\, \neg copy'\ y\ F') \wedge \\
& & \qquad \forall(\neg copy'\ x\ (lam\ F_0) \wedge \neg copy'\ x\ (app\ F_1\ F_2) \wedge \neg copy' x\ x \wedge \\
& & \qquad \neg copy'\ (lam\ F'_0)\ y \wedge \neg copy'\ (app\ F'_1\ F'_2)\ y \wedge \neg copy'\ y\ y) \\
& & \qquad\quad \to copy'\ x\ y \to copy'\ (E\ x)\ (F\ y)).
\end{array}$$

Note that by static analysis of `copy'`, we know that x will never end up in the second argument of `copy'` and symmetrically this applies to y, too. Thus the call to $\mathrm{Not}^C_x, \mathrm{Not}^C_y$ are, in this case, useless; however since this kind of data flow analysis is in general undecidable, the augmentation procedure inserts the negation of a clause for every 'odd' position relevant to the pivot parameter. Now, consider the typing clause for lambda terms:

$$\begin{array}{rcl}
oflam & : & \forall E : exp \to exp.\, \forall T_1, T_2 : tp. \\
& & of\ (lam\ E)\ (arrow\ T_1\ T_2) \\
& & \quad \leftarrow (\forall x : exp.\, of\ x\ T_1 \to of\ (E\ x)\ T_2).
\end{array}$$

As T_1 is an input term, normalization of input variables inserts the appropriate equation, where **oflam** $\overset{D}{\mapsto}$ **oflam'**:

$$\begin{array}{rcl}
oflam' & : & \forall E : exp \to exp.\, \forall T_1, T_2 : tp. \\
& & of\ (lam\ E)\ (arrow\ T_1\ T_2) \\
& & \qquad (\forall T : tp.\, of\ x\ T \leftarrow T \doteq T_1) \to \\
& & \qquad\quad of\ (E\ x)\ T_2).
\end{array}$$

Both Not^C_x and Not^1_x generate no contribution; in particular $x{:}exp \vdash \mathrm{Not}^1_x(\forall T : tp.\, of\ x\ T)$ calls the Not^i_x rule, but term complementation (applied to the existential variable T) yields the trivial clause. Therefore, by $\mathrm{Not}_{\mathrm{G}}(T \doteq T_1)$, augmentation will result into:

$$\begin{array}{rcl}
aug_{\mathrm{D}}(oflam) & : & \forall E : exp \to exp.\, \forall T_1, T_2 : tp. \\
& & of\ (lam\ E)\ (arrow\ T_1\ T_2) \\
& & \quad \leftarrow (\forall x : exp.\, of\ x\ T_1 \to \\
& & \qquad (\forall T : tp.\, \neg of\ x\ T \leftarrow T \not\doteq T_1) \to \\
& & \qquad\quad of\ (E\ x)\ T_2).
\end{array}$$

Consider now a predicate which counts the number of bound variables in a lambda-term:

$$\begin{array}{rcl}
cntlam & : & \forall E : exp \to exp.\, \forall N : nat. \\
& & cnt\ (lam\ E)\ N \\
& & \quad \leftarrow (\forall x : exp.\, cnt\ x\ s(0) \to \\
& & \qquad cnt\ (E\ x)\ N).
\end{array}$$

The call to $x{:}exp \vdash \mathrm{Not}_\alpha(cnt\ x\ s(0))$ leads to $\mathrm{Not}^1_x(cnt\ x\ s(0))$, since, again, Not^C_x does not contribute: Not^i_x collects the term complement of $s(0)$ in the *empty* context, that is $\{0, s(s(M))\}$:

$$\begin{array}{lll} aug_{\mathrm{D}}(cntlam) & : & \forall E\!:\!exp \to exp.\forall N\!:\!nat. \\ & & cnt\ (lam\ E)\ N \\ & & \quad \leftarrow (\forall x\!:\!exp.(\neg cnt\ x\ 0 \wedge \forall M\!:\!nat.\neg cnt\ x\ s(s(M))) \\ & & \qquad \to cnt\ (E\ x)\ N). \end{array}$$

Let us see how rule $\mathrm{Not}_\alpha\top$ enters the picture: recall the $\top$-normalized `linx` lambda clause:

$$\begin{array}{lll} linxlam & : & \forall E\!:\!exp \to exp \to exp. \\ & & linx\ (\lambda x\,.\,lam(\lambda y\,.\,E\ x\ y)) \\ & & \quad \leftarrow (\forall z\!:\!exp.\top_{linx} \to linx\ (\lambda x\,.\,E\ x\ z)). \end{array}$$

The judgment $z{:}exp; \top_{linx} \vdash aug_{\mathrm{G}}(linx\ E\ x\ z)$ triggers the rule $\mathrm{Not}_\alpha\top$; in turn $\cdot \vdash sh(z, exp \to exp) = \lambda y\,.\,z$ and thus $\mathrm{Not}^z_1(\top_{linx}) = \neg linx\ (\lambda x\,.\,z)$:

$$\begin{array}{lll} aug_{\mathrm{D}}(linxlam) & : & \forall E\!:\!exp \to exp \to exp. \\ & & linx\ (\lambda x\,.\,lam(\lambda y\,.\,E\ x\ y)) \\ & & \quad \leftarrow (\forall z\!:\!exp.\neg linx\ (\lambda x\,.\,z) \to linx\ (\lambda x\,.\,E\ x\ z)). \end{array}$$

Example 1.27 *Let us apply the complement algorithm to the* `linx` *predicate definition; note the vacuous application* $E_2\ x^0$*:*

$$\begin{array}{rcl} linxx & : & linx(\lambda x\,.\,x). \\ linxap1 & : & linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x^0)) \leftarrow linx(\lambda x\,.\,E_1\ x). \\ linxap2 & : & linx(\lambda x\,.\,app\ (E_1\ x^0)\ (E_2\ x)) \leftarrow linx(\lambda x\,.\,E_2\ x). \\ aug_{\mathrm{D}}(linxlm) & : & linx(\lambda x\,.\,lam(\lambda y\,.\,E\ x\ y)) \leftarrow (\forall y\!:\!exp.\top_{linx} \to linx(\lambda x\,.\,E\ x\ y)). \end{array}$$

$$\begin{array}{l} \mathrm{Not_D}(def(linx)) = \\ \mathrm{Not_D}(linxx) \vee \mathrm{Not_D}(linxap1) \vee \mathrm{Not_D}(linxap2) \vee \mathrm{Not_D}(linxlm) = \\ \quad (\neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x)) \wedge \neg linx(\lambda x\,.\,lam(\lambda y\,.\,(E\ x\ y)))) \vee \\ \quad (\neg linx(\lambda x\,.\,x) \wedge \neg linx(\lambda x\,.\,lam(\lambda y\,.\,(E\ x\ y))) \wedge \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x^1)) \\ \quad \wedge \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x)) \leftarrow \neg linx(\lambda x\,.\,E_1\ x)) \vee \\ \quad (\neg linx(\lambda x\,.\,x) \wedge \neg linx(\lambda x\,.\,lam(\lambda y\,.\,(E\ x\ y))) \wedge \neg linx(\lambda x\,.\,app\ (E_1\ x^1)\ (E_2\ x)) \\ \quad \leftarrow \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x)) \leftarrow \neg linx(\lambda x\,.\,E_2\ x)) \vee \\ \quad (\neg linx(\lambda x\,.\,x) \wedge \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x)) \\ \quad \wedge \neg linx(\lambda x\,.\,lam(\lambda y\,.\,(E\ x\ y))) \leftarrow (\forall y\!:\!exp.\neg linx(\lambda x\,.\,y) \to \neg linx(\lambda x\,.\,E\ x\ y))). \end{array}$$

We prove that augmented clauses and goals satisfy the augmented schema, where we define $\Gamma; aug_{\mathrm{D}}(\mathcal{D})$ as $\Gamma; \mathcal{D} \wedge \neg\mathcal{D}$, for $\Gamma \vdash \mathrm{Not}_\alpha(\mathcal{D}) = \neg\mathcal{D}$ and

$$\frac{}{aug(\circ) = \circ}\ aug\circ$$

$$\frac{aug(\mathcal{S}) = \mathcal{S}^a \qquad aug_{\mathrm{D}}(\mathcal{D}) = \mathcal{D}^a}{aug(\mathcal{S} \| (\Gamma; \mathrm{SOME}\ \Phi.\mathcal{D})) = \mathcal{S}^a \| (\Gamma; \mathrm{SOME}\ \Phi.\mathcal{D}^a)}\ aug\|$$

Now, the usual lemma on schema membership:

Lemma 1.28 *If* $\Gamma \vdash \Gamma'; \mathcal{D}' \in \mathcal{S}$ *then* $\Gamma \vdash \Gamma'; aug_{\mathrm{D}}(\mathcal{D}') \in aug(\mathcal{S})$.

Proof: By induction on the structure of $\pi :: \Gamma; \mathcal{D} \in \mathcal{S}$.

Case:

$$\frac{\Gamma \vdash \theta : \Phi \qquad (\Gamma'; \mathcal{D}') \equiv_\alpha (\Gamma''; \theta\mathcal{D})}{\Gamma \vdash (\Gamma'; \mathcal{D}') \in \mathcal{S} || (\Gamma''; \text{SOME } \Phi. \mathcal{D})} \in_1$$

$\Gamma'; \mathcal{D}' \equiv_\alpha \Gamma''; [\theta]\mathcal{D}$	By sub-derivation
$\Gamma'; aug_{\mathrm{D}}(\mathcal{D}') \equiv_\alpha \Gamma''; [\theta] aug_{\mathrm{D}}(\mathcal{D})$	By definition
$\Gamma'; aug_{\mathrm{D}}(\mathcal{D}') \in aug(\mathcal{S}) || \Gamma''; \text{SOME } \Phi.\, aug_{\mathrm{D}}(\mathcal{D})$	By rule $\in_1$
$\Gamma'; aug_{\mathrm{D}}(\mathcal{D}') \in aug(\mathcal{S} || \Gamma''; \text{SOME } \Phi. \mathcal{D})$	By rule $aug||$

Case: π ends in $\in_2$: by IH.

□

We use $\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D)$ for $\vdash aug_{\mathrm{D}}(D) = D^a$ and $\models_{\mathcal{S}^a} D^a$; analogously for $aug_{\mathrm{G}}(G)$.

Lemma 1.29 (Schema Augmentation) *Let $\Gamma; \mathcal{D} < \mathcal{S}$ and $aug(S) = \mathcal{S}^a$. Then:*

1. *If $\models_{\mathcal{S}} D$, then $\models_{\mathcal{S}^a} aug_{D}(D)$.*
2. *If $\Gamma; \mathcal{D} \setminus G < \mathcal{S}$, then $\Gamma; \mathcal{D} \setminus aug_{G}(G) < \mathcal{S}^a$.*
3. *If $\Gamma; \mathcal{D} < \mathcal{S}$ then $\Gamma; aug_{D}(\mathcal{D}) < \mathcal{S}^a$.*

Proof: By mutual induction on the structure of $\pi :: \models_{\mathcal{S}} D$, $\gamma :: \Gamma; \mathcal{D} \setminus G < \mathcal{S}$ and $\sigma :: \Gamma; \mathcal{D} < \mathcal{S}$. We show the crucial cases:

Case:

$$\pi = \frac{X \in \{Q, \top, \bot\}}{\models_{\mathcal{S}} X} \models_{\mathcal{S}} X$$

Immediately $\Gamma; \mathcal{D} \vdash aug_{\mathrm{D}}(X) = X$ and

$$\pi' = \frac{X \in \ \{Q, \top, \bot\}}{\models_{\mathcal{S}^a} X} \models_{\mathcal{S}} X$$

Case:

$$\pi = \frac{\cdot; \top \setminus [\rho] G < \mathcal{S}}{\models_{\mathcal{S}} \forall (G \to Q)} \models_{\mathcal{S}} \to^{\sigma}$$

$\cdot; \top \setminus [\rho] G < \mathcal{S}$	By sub-derivation
$\cdot; \top \setminus aug_{\mathrm{G}}([\rho]G) < \mathcal{S}^a$	By IH 2
$\models_{\mathcal{S}^a} aug_{\mathrm{G}}([\rho]G) \to [\rho]Q$	By rule $\models_{\mathcal{S}} \to$
$\models_{\mathcal{S}^a} aug_{\mathrm{D}}(\forall (G \to Q))$	By rule $aug_{\mathrm{D}} \to$

Case:

$$\gamma = \frac{X \in \{\top, \bot\} \text{ or (dis)eq} \qquad \Gamma; \mathcal{D} < \mathcal{S}}{\Gamma; \mathcal{D} \setminus X < \mathcal{S}} \setminus X$$

$\Gamma; \mathcal{D} < \mathcal{S}$	By sub-derivation
$\Gamma; aug_{\mathrm{D}}(\mathcal{D}) < \mathcal{S}^a$	By IH 3
$\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(X) = X$	By rule $aug_{\mathrm{G}} X$
$\Gamma; aug_{\mathrm{D}}(\mathcal{D}) \setminus X < \mathcal{S}^a$	By rule $\setminus X$

Case:

$$\gamma = \frac{\Gamma; \mathcal{D} < \mathcal{S}}{\Gamma; \mathcal{D} \setminus Q < \mathcal{S}} \setminus\mathrm{At}$$

By sub-derivation $\Gamma; \mathcal{D} < \mathcal{S}$; there are two ways to augment an atom:

Subcase: $\cdot; \top \vdash aug_{\mathrm{G}}(Q) = Q$:

$\cdot; \top < \mathcal{S}^a$	By rule $<_1$
$\cdot; \top \setminus Q < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\mathrm{At}\top$
$\cdot; \top \setminus aug_{\mathrm{G}}(Q) < \mathcal{S}^a$	By rule $\setminus\mathrm{At}$

Subcase: $\Gamma; \mathcal{D} \vdash aug(Q) = \mathrm{Not}_{\alpha}(\mathcal{D}) \rightarrow Q$:

$\Gamma; aug_{\mathrm{D}}(\mathcal{D}) < \mathcal{S}^a$	By IH 3
$\Gamma; aug_{\mathrm{D}}(\mathcal{D}) \setminus Q < \mathcal{S}^a$	By rule $\setminus\mathrm{At}$
$\Gamma; \mathcal{D} \wedge \mathrm{Not}_{\alpha}(\mathcal{D}) \setminus Q < \mathcal{S}^a$	By def. of $aug_{\mathrm{D}}(\mathcal{D})$
$\Gamma; \mathcal{D} \setminus\ \mathrm{Not}_{\alpha}(\mathcal{D}) \rightarrow Q < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\mathrm{At}$
$\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(Q) < \mathcal{S}^a$	By rule $\setminus \rightarrow$

Case:

$$\gamma = \frac{\models_{\mathcal{S}} D \qquad \Gamma; \mathcal{D} \wedge D \setminus G < \mathcal{S}}{\Gamma; \mathcal{D} \setminus D \rightarrow G < \mathcal{S}} \setminus \rightarrow$$

$\models_{\mathcal{S}} D$	By sub-derivation
$\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D)$	By IH 1
$\Gamma; \mathcal{D} \wedge D \setminus G < \mathcal{S}$	By sub-derivation
$\Gamma; \mathcal{D} \wedge D \setminus aug_{\mathrm{G}}(G) < \mathcal{S}^a$	By IH 2
$\Gamma; \mathcal{D} \setminus D \rightarrow aug_{\mathrm{G}}(G) < \mathcal{S}^a$	By rule $\setminus \rightarrow$
$\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(D \rightarrow G) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}} \rightarrow$

Case:

$$\sigma = \frac{}{\cdot; \top < \mathcal{S}} <_1$$

Then immediately:

$$\sigma' = \frac{}{\cdot; \top < \mathcal{S}^a} <_1$$

Case:

$$\sigma = \frac{\Gamma \vdash (\Gamma'; \mathcal{D}') \in \mathcal{S} \quad \models_{\mathcal{S}} \mathcal{D}' \quad (\Gamma; \mathcal{D}) < \mathcal{S}}{(\Gamma, \lceil \Gamma' \rceil); (\mathcal{D} \wedge \lceil \mathcal{D}' \rceil) < \mathcal{S}} <_2$$

$\Gamma; \mathcal{D} < \mathcal{S}$	By sub-derivation
$\Gamma; aug_{\mathrm{D}}(\mathcal{D}) < \mathcal{S}^a$	By IH 3
$\Gamma \vdash \Gamma'; \mathcal{D}' \in \mathcal{S}$	By sub-derivation
$\Gamma'; aug_{\mathrm{D}}(\mathcal{D}') \in \mathcal{S}^a$	By Lemma 6.28
$\models_{\mathcal{S}} \mathcal{D}'$	By sub-derivation
$\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D')$	By IH 1
$(\Gamma, \lceil \Gamma' \rceil); aug_{\mathrm{D}}(\mathcal{D}) \wedge aug_{\mathrm{D}}(\lceil \mathcal{D}')\rceil) < \mathcal{S}^a$	By rule $<_2$
$(\Gamma, \lceil \Gamma' \rceil); aug_{\mathrm{D}}(\mathcal{D} \wedge \lceil \mathcal{D}' \rceil) < \mathcal{S}^a$	By rule $aug_{\mathrm{D}}\wedge$

□

The following Lemma ensures that augmented clauses are closed under negation. In particular, for $\Gamma; \mathcal{D} < \mathcal{S}^a$, if $D \in \mathcal{D}$, so is $\Gamma \vdash \mathrm{Not}_{\alpha}(D)$. We use $\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\mathrm{G}}(G) < \mathcal{S}^a$ for $\mathrm{Not}_{\mathrm{G}}(G) = G_{\neg}$ and $\Gamma; \mathcal{D} \setminus G_{\neg} < \mathcal{S}^a$. Similarly for $\mathrm{Not}_{\mathrm{D}}(D)$. We will need the following technical remark:

Remark 1.30 *If $\vec{M}$ is a simple term, then* $\models_{\mathcal{S}^a} \bigwedge_{\vec{N} \in \mathrm{Not}(\vec{M})} \forall(\neg(q\ \vec{N}) \leftarrow \top)$.

Proof: By rules $\models_{\mathcal{S}^a}\wedge, \models_{\mathcal{S}}\forall, \models_{\mathcal{S}^a}\!\rightarrow, \backslash\top, \models_{\mathcal{S}}\mathrm{At}, \backslash\mathrm{At}$. $\square$

Lemma 1.31 (Negative Schema Augmentation) *Let* $\Gamma; \mathcal{D} < \mathcal{S}^a$. *Then:*

1. *If* $\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D)$, *then* $\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D))$.
2. *If* $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G) < \mathcal{S}^a$, *then* $\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}(G)) < \mathcal{S}^a$.

Proof: By mutual induction on the structure of $\pi :: \models_{\mathcal{S}^a} aug_{\mathrm{D}}(D)$ and $\gamma :: \Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G) < \mathcal{S}^a$.

Case:

$$\pi = \frac{}{\models_{\mathcal{S}^a} aug_{\mathrm{D}}(\bot)} \models_{\mathcal{S}^a}\bot$$

By rule $aug_{\mathrm{D}}(\bot) = \bot$ and $\vdash \mathrm{Not}_{\mathrm{D}}(\bot) = \top$: thus

$$\pi' = \frac{}{\models_{\mathcal{S}^a} \top} \models_{\mathcal{S}^a}\top$$

Case:

$$\pi = \frac{}{\models_{\mathcal{S}^a} aug_{\mathrm{D}}(\top)} \models_{\mathcal{S}^a}\top$$

By rule $aug_{\mathrm{D}}(\top) = \top$ and $\vdash \mathrm{Not}_{\mathrm{D}}(\top) = \bot$: thus

$$\pi' = \frac{}{\models_{\mathcal{S}^a} \bot} \models_{\mathcal{S}^a}\bot$$

Case: π ends in $\models_{\mathcal{S}^a} aug_{\mathrm{D}}(\forall(q\ \vec{M} \leftarrow G))$:

$\models_{\mathcal{S}^a} (\forall(q\ \vec{M}) \leftarrow aug_{\mathrm{G}}(G))$	By rule $aug_{\mathrm{D}} \rightarrow$
$\cdot; \top \setminus aug_{\mathrm{G}}([\rho]G) < \mathcal{S}^a$	By inversion on rule $\models_{\mathcal{S}}\rightarrow$
$\cdot; \top \setminus \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}([\rho]G)) < \mathcal{S}^a$	By IH 2
$\models_{\mathcal{S}^a} [\rho](\neg q\ \vec{M})$	By rule $\models_{\mathcal{S}^a}\mathrm{At}$
$\models_{\mathcal{S}^a} \forall(\neg q\ \vec{M} \leftarrow \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}(G)))$	By rule $\models_{\mathcal{S}^a}\!\rightarrow$
$\models_{\mathcal{S}^a} \bigwedge_{\vec{N} \in \mathrm{Not}(\vec{M})} \forall(\neg(q\ \vec{N}))$	By Remark 6.30
$\models_{\mathcal{S}^a} \bigwedge_{\vec{N} \in \mathrm{Not}(\vec{M})} \forall(\neg(q\ \vec{N}) \leftarrow \top) \wedge \forall(\neg q\ \vec{M} \leftarrow \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}(G)))$	By rule $\models_{\mathcal{S}^a}\wedge$
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(\forall(q\ \vec{M}) \leftarrow aug_{\mathrm{G}}(G))$	By rule $\mathrm{Not}_{\mathrm{D}} \rightarrow$
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(\forall(q\ \vec{M} \leftarrow G)))$	By rule $aug_{\mathrm{D}} \rightarrow$

Case: π ends in $\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D_1 \wedge D_2)$:

$\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D_1) \wedge aug_{\mathrm{D}}(D_2)$	By rule $aug_{\mathrm{D}}\wedge$
$\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D_1)$ and $\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D_2)$	By rule $\models_{\mathcal{S}}\wedge$
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_1))$ and $\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_2))$	By IH 1
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_1)) \vee \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_2))$	By rule $\models_{\mathcal{S}}\vee$
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_1) \wedge aug_{\mathrm{D}}(D_2))$	By rule $\mathrm{Not}_{\mathrm{D}}\wedge$
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_1 \wedge D_2))$	By rule $aug_{\mathrm{D}}\wedge$

Case: π ends in $\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D_1 \vee D_2)$:

$\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D_1) \vee aug_{\mathrm{D}}(D_2)$	By rule $aug_{\mathrm{D}}\vee$
$\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D_1)$ and $\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D_2)$	By rule $\models_{\mathcal{S}}\vee$
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_1))$ and $\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_2))$	By IH 1
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_1)) \wedge \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_2))$	By rule $\models_{\mathcal{S}}\wedge$
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_1) \vee aug_{\mathrm{D}}(D_2))$	By rule $\mathrm{Not}_{\mathrm{D}}\vee$
$\models_{\mathcal{S}^a} \mathrm{Not}_{\mathrm{D}}(aug_{\mathrm{D}}(D_1 \vee D_2))$	By rule $aug_{\mathrm{D}}\vee$

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(\top) < \mathcal{S}^a$:

$\Gamma; \mathcal{D} \setminus \top < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\top$
$\Gamma; \mathcal{D} < \mathcal{S}^a$	By sub-derivation
$\Gamma; \mathcal{D} \vdash \mathrm{Not}_{\mathrm{G}}(\top) = \bot$	By rule $\mathrm{Not}_{\mathrm{G}}\top$
$\Gamma; \mathcal{D} \setminus \bot < \mathcal{S}^a$	By rule $\setminus\bot$

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(\bot) < \mathcal{S}^a$:

$\Gamma; \mathcal{D} \setminus \bot < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\bot$
$\Gamma; \mathcal{D} < \mathcal{S}^a$	By sub-derivation
$\Gamma; \mathcal{D} \vdash \mathrm{Not}_{\mathrm{G}}(\bot) = \top$	By rule $\mathrm{Not}_{\mathrm{G}}\bot$
$\Gamma; \mathcal{D} \setminus \top < \mathcal{S}^a$	By rule $\setminus\bot$

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(\vec{M} \doteq \vec{N}) < \mathcal{S}^a$:

$\Gamma; \mathcal{D} \setminus \vec{M} \doteq \vec{N} < \mathcal{S}^a$	By rule $aug_{\mathrm{G}} \doteq$
$\Gamma; \mathcal{D} < \mathcal{S}^a$	By sub-derivation
$\Gamma; \mathcal{D} \vdash \mathrm{Not}_{\mathrm{G}}(\vec{M} \doteq \vec{N}) = (\vec{M} \not\doteq \vec{N})$	By rule $\mathrm{Not}_{\mathrm{G}} \doteq$
$\Gamma; \mathcal{D} \setminus \vec{M} \not\doteq \vec{N} < \mathcal{S}^a$	By rule $\setminus \not\doteq$

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(\vec{M} \not\doteq \vec{N}) < \mathcal{S}^a$:

$\Gamma; \mathcal{D} \setminus \vec{M} \not\doteq \vec{N} < \mathcal{S}^a$	By rule $aug_{\mathrm{G}} \not\doteq$
$\Gamma; \mathcal{D} < \mathcal{S}^a$	By sub-derivation
$\Gamma; \mathcal{D} \vdash \mathrm{Not}_{\mathrm{G}}(\vec{M} \not\doteq \vec{N}) = (\vec{M} \doteq \vec{N})$	By rule $\mathrm{Not}_{\mathrm{G}} \not\doteq$
$\Gamma; \mathcal{D} \setminus \vec{M} \doteq \vec{N} < \mathcal{S}^a$	By rule $\setminus \not\doteq$

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(Q) < \mathcal{S}^a$:
By sub-derivation $\Gamma; \mathcal{D} < \mathcal{S}^a$; there are two ways to augment an atom:

Subcase: $\cdot; \top \vdash aug_{\mathrm{G}}(Q) = Q$:

$\cdot; \top \setminus\ Q < \mathcal{S}^a$	By rule $\setminus$At
$\cdot; \top \setminus \neg Q < \mathcal{S}^a$	By rule $\mathrm{Not}_{\mathrm{G}}$At
$\cdot; \top \setminus \mathrm{Not}_{\mathrm{G}}(Q) < \mathcal{S}^a$	By rule $\mathrm{Not}_{\mathrm{G}} \rightarrow$
$\cdot; \top \setminus \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}(Q)) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\mathrm{At}\top$

Subcase: $\Gamma; \mathcal{D} \vdash alg(Q) = \mathrm{Not}_{\alpha}(\mathcal{D}) \rightarrow G$:

$\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\alpha}(\mathcal{D}) \rightarrow Q < \mathcal{S}^a$	By rule aug_{G}At
$\Gamma; (\mathcal{D} \wedge \mathrm{Not}_{\alpha}(\mathcal{D})) \setminus Q < \mathcal{S}^a$	By rule $\setminus \rightarrow$
$\Gamma; (\mathcal{D} \wedge \mathrm{Not}_{\alpha}(\mathcal{D})) < \mathcal{S}^a$	By sub-derivation
$\Gamma; (\mathcal{D} \wedge \mathrm{Not}_{\alpha}(\mathcal{D})) \setminus \neg Q < \mathcal{S}^a$	By rule $\setminus$At
$\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\alpha}(\mathcal{D}) \rightarrow \neg Q < \mathcal{S}^a$	By rule $\setminus \rightarrow$
$\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\alpha}(\mathcal{D}) \rightarrow \mathrm{Not}_{\mathrm{G}}(Q) < \mathcal{S}^a$	By rule $\mathrm{Not}_{\mathrm{G}}$At
$\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\mathrm{G}}(\mathrm{Not}_{\alpha}(\mathcal{D}) \rightarrow Q) < \mathcal{S}^a$	By rule $\mathrm{Not}_{\mathrm{G}} \rightarrow$
$\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}(Q)) < \mathcal{S}^a$	By rule aug_{G}At

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(D \rightarrow G) < \mathcal{S}^a$:

$\Gamma; \mathcal{D} \setminus aug_{\mathrm{D}}(D) \rightarrow aug_{\mathrm{G}}(G) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}} \rightarrow$
$\models_{\mathcal{S}^a} aug_{\mathrm{D}}(D)$ and $\Gamma; (\mathcal{D} \wedge D) \setminus aug_{\mathrm{G}}(G) < \mathcal{S}^a$	By rule $\setminus \rightarrow$
$\Gamma; (\mathcal{D} \wedge aug_{\mathrm{D}}(D)) \setminus \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}(G)) < \mathcal{S}^a$	By IH 2
$\Gamma; \mathcal{D} \setminus aug_{\mathrm{D}}(D) \rightarrow \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}(G)) < \mathcal{S}^a$	By rule $\setminus \rightarrow$
$\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{D}}(D) \rightarrow aug_{\mathrm{G}}(G)) < \mathcal{S}^a$	By rule $\mathrm{Not}_{\mathrm{G}} \rightarrow$
$\Gamma; \mathcal{D} \setminus \mathrm{Not}_{\mathrm{G}}(aug_{\mathrm{G}}(D \rightarrow G)) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}} \rightarrow$

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(\forall x\!:\!a.\, G) < \mathcal{S}^a$:

$\Gamma; \mathcal{D} \setminus \forall x\!:\!a.\, aug_{\mathrm{G}}(G) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\forall$
$(\Gamma, y{:}A); \mathcal{D} \setminus [y/x]aug_{\mathrm{G}}(G) < \mathcal{S}^a$	By rule $\setminus\forall$
$(\Gamma, y{:}A); \mathcal{D} \setminus [y/x]\mathrm{Not_G}(aug_{\mathrm{G}}(G)) < \mathcal{S}^a$	By IH 2
$\Gamma; \mathcal{D} \setminus \forall x\!:\!a.\, \mathrm{Not_G}(aug_{\mathrm{G}}(G)) < \mathcal{S}^a$	By rule $\setminus\forall$
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(\forall x\!:\!a.\, aug_{\mathrm{G}}(G)) < \mathcal{S}^a$	By rule $\mathrm{Not_G}\forall$
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(\forall x\!:\!a.\, G)) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\forall$

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G_1 \vee G_2) < \mathcal{S}^a$:

$\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G_1) \vee aug_{\mathrm{G}}(G_2) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\vee$
$\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G_1) < \mathcal{S}^a$ and $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G_2) < \mathcal{S}^a$	By rule $\setminus\vee$
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_1)) < \mathcal{S}^a$ and $\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_2)) < \mathcal{S}^a$	By IH 2
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_1)) \wedge \mathrm{Not_G}(aug_{\mathrm{G}}(G_2) < \mathcal{S}^a$	By rule $\setminus\wedge$
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_1) \vee aug_{\mathrm{G}}(G_2)) < \mathcal{S}^a$	By rule $\mathrm{Not_G}\vee$
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_1 \vee G_2)) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\vee$

Case: γ ends in $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G_1 \wedge G_2) < \mathcal{S}^a$:

$\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G_1) \wedge aug_{\mathrm{G}}(G_2) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\wedge$
$\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G_1) < \mathcal{S}^a$ and $\Gamma; \mathcal{D} \setminus aug_{\mathrm{G}}(G_2) < \mathcal{S}^a$	By rule $\setminus\wedge$
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_1)) < \mathcal{S}^a$ and $\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_2)) < \mathcal{S}^a$	By IH 2
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_1)) \vee \mathrm{Not_G}(aug_{\mathrm{G}}(G_2) < \mathcal{S}^a$	By rule $\setminus\vee$
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_1) \wedge aug_{\mathrm{G}}(G_2)) < \mathcal{S}^a$	By rule $\mathrm{Not_G}\wedge$
$\Gamma; \mathcal{D} \setminus \mathrm{Not_G}(aug_{\mathrm{G}}(G_1 \wedge G_2)) < \mathcal{S}^a$	By rule $aug_{\mathrm{G}}\wedge$

□

We can combine the two latter lemmata 6.29 and 6.31 to prove that augmentation guarantees that static and dynamic clauses are closed under complementation.

Corollary 1.32 (Closure under Complementation)
If $\models_{\mathcal{S}} D$, *then* $\models_{\mathcal{S}^a} \mathrm{Not_D}(aug_{\mathrm{D}}(D))$.

1.7 Exclusivity

We are now in the position to prove the main result of this Chapter, namely that clause complementation satisfies the boolean rules of negation, in the form of exclusivity and exhaustivity. We remark that this holds due to the fact that context schemata allow to pose only 'well-behaved' goals. For example, consider the query $G \equiv \neg even(0) \rightarrow even(s(s(0)))$, which is such that both $\cdot; \top \vdash_{even} G$ and $\cdot; \top \vdash_{\neg even} \mathrm{Not_G}(G)$ are provable. By definition of schema satisfaction, this is not a legal query. Similarly, the following counter-example to exhaustivity $\cdot; \top \vdash_{even} \forall x : nat.\, even(x)$ is not allowed. Moreover, the Context Preservation Theorems (Theorem 6.13 and 6.14) guarantees that, from allowed run-time contexts and goals, only allowed subgoals are generated.

We will need the following obvious fact:

Lemma 1.33 *If* $\Gamma \vdash M : A$ *and* M *is rigid, so is every* N *such that* $N \in (\Gamma \vdash \mathrm{Not}(M))$.

Proof: By a straightforward induction on the structure of $\Gamma \vdash \mathrm{Not}(M)$. □

We use $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G)$ for $\Gamma \vdash \mathrm{Not_G}(G) = G'$ and $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G'$; also, $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D) \gg \neg Q$ for $\mathrm{Not_D}(D) = D'$ and $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D' \gg \neg Q$; similarly for Not_{α}. We use $\mathcal{P}$ to denote the conjunction of a positive program $\mathcal{P}^+$ and its negation $\mathcal{P}^- = \mathrm{Not_D}(\mathcal{P}^+)$.

Theorem 1.34 (Exclusivity) *Let* $\models_{\mathcal{S}^a} aug_D(\mathcal{P})$ *and* $\Gamma; \mathcal{D} < \mathcal{S}^a$*. For every goals such that* $\Gamma; \mathcal{D} \setminus G < \mathcal{S}^a$*, it is not the case that both* $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G$ *and* $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G)$*.*

Proof: We generalize this to;

$\boxed{1}$ It is not the case that there is $\mathcal{S}^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} G$ and there is $\mathcal{S}^- :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G)$.

$\boxed{2.1}$ Let $D \sqsubseteq \mathrm{def}(q, \mathcal{P})$: It is not the case that $\mathcal{I}^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ and $\mathcal{I}^- :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D) \gg \mathrm{Not_G}(Q)$. By mutual induction on the structure of $\mathcal{I}^+$ and $\mathcal{S}^+$, by assuming their existence:

Case:

$$\mathcal{I}^+ = \frac{}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bot_q \gg Q} \gg\bot$$

But there can be no proof of $\neg Q$ from $\mathrm{Not_D}(\bot_q) = \top_q$.

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_n} \doteq \overline{M_n} \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(q\ \overline{N_n} \leftarrow G) \gg q\ \overline{M_n}} \gg\to$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(\forall(q\ \overline{N_n} \leftarrow G)) \gg \neg q\ \overline{M_n}$:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_n} \doteq \overline{M_n}$	By sub-derivation
$\cdot \vdash \overline{M_n} \in \lVert\overline{N_n}\rVert\ (*)$	By definition of of $\lVert\_\rVert$
$\mathcal{S}_1^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G$	By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{\overline{S_n} \in \cdot \vdash \mathrm{Not}(\overline{N_n})} \forall(\neg q\ \overline{S_n} \leftarrow \top) \wedge \forall(\neg q\ \overline{N_n} \leftarrow \mathrm{Not_G}(G)) \gg \neg q\ \overline{M_n}$	By rule $\mathrm{Not_D} \to$
Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{\overline{S_n} \in \cdot \vdash \mathrm{Not}(\overline{N_n})} \forall(\neg q\ \overline{S_n} \leftarrow \top) \gg \neg q\ \overline{M_n}$	By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(\neg q\ \overline{S_n} \leftarrow \top) \gg \neg q\ \overline{M_n}$ for some $\overline{S_n}$	By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta](\neg q\ \overline{S_n}) \leftarrow \top \gg \neg q\ \overline{M_n}$ for some θ	By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta]\overline{S_n} \doteq \overline{M_n}$	By inversion
$\cdot \vdash \overline{M_n} \in \lVert\mathrm{Not}(\overline{N_n})\rVert$	By definition of $\lVert\_\rVert$
$\bot$	By term exclusivity with $(*)$
Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(\neg q\ \overline{N_n} \leftarrow \mathrm{Not_G}(G)) \gg \neg q\ \overline{M_n}$	By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}([\sigma]G)$	By inversion
$\bot$	By IH 1 w.r.t. $\mathcal{S}_1^+$

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (D_1 \vee D_2) \gg Q} \gg\vee$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1 \vee D_2) \gg \neg Q$:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$	By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1) \wedge \mathrm{Not_D}(D_2) \gg \neg Q$	By rule $\mathrm{Not_D}\vee$
Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1) \gg \neg Q$	By inversion
$\bot$	By IH 2.1
Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_2) \gg \neg Q$	By inversion
$\bot$	By IH 2.1

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (D_1 \wedge D_2) \gg Q} \gg\wedge$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1 \wedge D_2) \gg \neg Q$:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$	By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1) \vee \mathrm{Not_D}(D_2) \gg \neg Q$	By rule $\mathrm{Not_D}\wedge$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1)$	By inversion
$\bot$	By IH 2.1

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (D_1 \wedge D_2) \gg Q} \gg\wedge$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1 \wedge D_2) \gg \neg Q$:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q$ — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1) \vee \mathrm{Not_D}(D_2) \gg \neg Q$ — By rule $\mathrm{Not_D}\wedge$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_2) \gg \neg Q$ — By inversion
$\perp$ — By IH 2.1

$\boxed{2.2}$ Let $D \sqsubseteq \mathrm{def}(q, \mathcal{P})$ and $D' \sqsubseteq \mathrm{def}(q, \mathcal{D})$. It is not the case that $\mathcal{I}^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ and $\mathcal{I}^- :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D') \gg \neg Q$. By simultaneous induction on the structure of $\mathcal{I}^+$ and $\mathcal{I}^-$:

No Case: $\mathcal{I}^+$ ends in $\gg\top$

Case:

$$\mathcal{I}^+ = \frac{}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \perp_q \gg Q} \gg\mathrm{At}$$

But by schema satisfaction there can be no formula in the dynamic definition of q.

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{S_n} \doteq \overline{M_n} \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(q\ \overline{S_n} \leftarrow G) \gg q\ \overline{M_n}} \gg\to$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(\forall(q\ \overline{T_n} \leftarrow H)) \gg \neg q\ \overline{M_n}$.

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{S_n} \doteq \overline{M_n}$ — By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]S_i \doteq M_i$ for every $1 \le i \le n$ $(*)$ — By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{x \in \mathrm{dom}(\Gamma)} \mathrm{Not}^C_x(q\ \overline{T_n}) \wedge (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}^i_x(q\ \overline{T_n}))) \wedge \forall(\neg q\ \overline{T_n}) \leftarrow \mathrm{Not_G}(G) \gg \neg q\ \overline{M_n}$
By rule $\mathrm{Not}_\alpha \to$

Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}^C_x(q\ \overline{T_n}) \gg \neg q\ \overline{M_n}$ — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le j \le n} \forall Z_1 : A_1. \ldots \forall Z_n : A_n. \neg q\ [e_x/Z_j]\overline{Z_n} \gg \neg q\ \overline{M_n}$ — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} e_x \doteq M_j$ — By inversion
$\perp$ — From line $(*)$ and S_j rigid

Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}^i_x(q\ \overline{T_n}) \gg \neg q\ \overline{M_n}$ for a fixed x — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}^i_x(q\ \overline{T_n}) \gg \neg q\ \overline{M_n}$ — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le j \le n} \bigwedge_{N \in \Gamma \vdash \mathrm{Not}(T_j)} (\forall Z_1 : A_1. \ldots \forall Z_n : A_n. \neg q\ [e_x/Z_i, N/Z_j]\overline{Z_n}) \gg \neg q\ \overline{M_n}$
By rule $\mathrm{Not}^i_x \to$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} e_x \doteq M_i$ — By inversion
$\perp$ — From line $(*)$ and S_i rigid

Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(\neg q\ \overline{T_n} \leftarrow \mathrm{Not_G}(H)) \gg \neg q\ \overline{M_n}$ — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta]\overline{T_n} \doteq \overline{M_n}$ — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} e_x \doteq M_j$ for some $1 \le j \le n$ — By complementable clause
$\perp$ — From line $(*)$ and S_j rigid

Case: $\mathcal{I}^+$ ends in $\gg\to$, $\mathcal{I}^-$ ends in $\gg\perp$: But there can be no proof of $\neg Q$ from $\Gamma \vdash \mathrm{Not}_\alpha(\perp_q) = \top_q$.

Case: $\mathcal{I}^+$ ends in $\gg\to$, $\mathcal{I}^-$ ends in $\gg\top$:

Subcase: $\Gamma = \cdot$ or for all $x \in \mathrm{dom}(\Gamma)$, for all i, $1 \le i \le n$ it is not the case that xR^iq: then $\Gamma \vdash \mathrm{Not}_\alpha(\top_q) = \top_q$ and an be no proof of $\neg Q$ from $\top_q$.

Subcase: else $\Gamma \vdash \mathrm{Not}_\alpha(\top_q) = \bigwedge_{x \in dom(\Gamma)} (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}^C_x(\top_q))$; fix x, i:
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}^C_x(\top_q) \gg \neg q\ \overline{M_n}$ — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le i \le n} \forall Z_1 : A_1. \ldots \forall Z_n : A_n. \neg q\ [e_x/Z_j]\overline{Z_n} \gg \neg q\ \overline{M_n}$ — By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} e_x \doteq M_j$ — By inversion
$\perp$ — Above line and rigidity restriction on S_j $(*)$.

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (D_1 \vee D_2) \gg Q} \gg\vee$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash \mathrm{Not}_\alpha(D') \gg \neg Q$:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$ — By sub-derivation
$\perp$ — By IH 2.2

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (D_1 \wedge D_2) \gg Q} \gg\wedge_1$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash \mathrm{Not}_\alpha(D') \gg \neg Q$:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$ — By sub-derivation
$\perp$ — By IH 2.2

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (D_1 \wedge D_2) \gg Q} \gg\wedge_2$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash \mathrm{Not}_\alpha(D') \gg \neg Q$: Symmetric to the above.

Case:

$$\mathcal{I}^- = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D'_1) \gg \neg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D'_1 \vee D'_2) \gg \neg Q} \gg\wedge_1$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash D \gg Q$. By IH 2.2

Case:

$$\mathcal{I}^- = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D'_2) \gg \neg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D'_1 \vee D'_2) \gg \neg Q} \gg\wedge_2$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash D \gg Q$. By IH 2.2.

Case:

$$\mathcal{I}^- = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D'_1) \gg \neg Q \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D'_2) \gg \neg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D'_1 \wedge D'_2) \gg \neg Q} \gg\vee$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash D \gg Q$. By IH 2.2.

$\boxed{2.3}$ Let $D \sqsubseteq \mathrm{def}(q, \mathcal{D})$ and $D' \sqsubseteq \mathrm{def}(q, \mathcal{P})$. It is not the case that $\mathcal{I}^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ and $\mathcal{I}^- :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{D}}(D') \gg \neg Q$. By simultaneous induction on the structure of $\mathcal{I}^+$ and $\mathcal{I}^-$. Note that by the restriction to complementable programs, some SPE e_x occurs in D; we thus use the notation $\overline{N^i_{e_x}}$.

No Case: for $\mathcal{I}^+$ ends in $\gg\top$.

No Case: for $\mathcal{I}^+$ ends in $\gg\perp$, since by the restriction to complementable clauses, $\perp$ cannot occur as an assumption.

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N^i_{e_x}} \doteq \overline{M_n} \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]H}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(q\ \overline{N^i_{e_x}} \leftarrow H) \gg q\ \overline{M_n}} \gg\rightarrow$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{D}}(\forall(q\ \overline{S_n} \leftarrow G)) \gg \neg q\ \overline{M_n}$.

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N^i_{e_x}} \doteq \overline{M_n}$ — By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} e_x \doteq M_i\ (*)$ for some $1 \leq i \leq n$ — By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{\overline{T_n} \in \cdot \vdash \mathrm{Not}(\overline{S_n})} \forall(\neg q\ \overline{T_n} \leftarrow \top) \wedge \forall(\neg q\ \overline{S_n} \leftarrow \mathrm{Not}_{\mathrm{G}}(G)) \gg \neg q\ \overline{M_n}$ — By rule $\mathrm{Not}_{\mathrm{D}} \rightarrow$

Subcase: $\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \bigwedge_{\overline{T_n}\in\cdot\vdash\mathrm{Not}(\overline{S_n})} \forall(\neg q\ \overline{T_n} \leftarrow \top)\gg q\ \overline{M_n}$ By inversion
$\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \forall(\neg q\ \overline{T_n} \leftarrow \top)\gg q\ \overline{M_n}$ for some $\overline{T_n}\in\cdot\vdash\mathrm{Not}(\overline{S_n})$ By inversion
$\Gamma;\mathcal{D}\vdash_{\mathcal{P}} [\theta](\neg q\ \overline{T_n} \leftarrow \top)\gg\neg q\ \overline{M_n}$ for some θ By inversion
$\Gamma;\mathcal{D}\vdash_{\mathcal{P}} [\theta]\overline{T_n} \doteq \overline{M_n}$ By inversion
$\Gamma;\mathcal{D}\vdash_{\mathcal{P}} [\theta]T_i \doteq M_i$ By inversion
$\bot$ $(*)$ and Lemma 6.33 on above line

Subcase: $\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \forall(\neg q\ \overline{S_n} \leftarrow \mathrm{Not_G}(G))\gg\neg q\ \overline{M_n}$ By inversion
$\Gamma;\mathcal{D}\vdash_{\mathcal{P}} [\theta']\overline{S_n} \doteq \overline{M_n}$ By inversion
$\Gamma;\mathcal{D}\vdash_{\mathcal{P}} [\theta']S_i \doteq M_i$ By inversion
$\bot$ From line $(*)$ and S_i rigid.

No Case: $\mathcal{I}^+$ ends in $\gg\ \rightarrow$, $\mathcal{I}^-$ ends in $\gg\bot$: if $\top_q$ is the only definition for q, then by schema satisfaction $\forall(q\ \overline{N^i_{e_x}} \leftarrow H)$ could not be an assumption.

Case: $\mathcal{I}^+$ ends in $\gg\rightarrow$, $\mathcal{I}^-$ ends in $\gg\bot$: ditto.

Case:

$$\mathcal{I}^+ = \frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} D_1\gg Q}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} (D_1\wedge D_2)\gg Q}\gg\wedge_1$$

and $\mathcal{I}^-$ ends in $\Gamma;\mathcal{D}\vdash \mathrm{Not_D}(D')\gg\neg Q$:
$\Gamma;\mathcal{D}\vdash_{\mathcal{P}} D_1\gg Q$ By sub-derivation
$\bot$ By IH 2.2

Case:

$$\mathcal{I}^+ = \frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} D_2\gg Q}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} (D_1\wedge D_2)\gg Q}\gg\wedge_2$$

and $\mathcal{I}^-$ ends in $\Gamma;\mathcal{D}\vdash \mathrm{Not_D}(D')\gg\neg Q$: Symmetric to the above.

Case:

$$\mathcal{I}^- = \frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not_D}(D'_1)\gg\neg Q}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not_D}(D'_1\vee D'_2)\gg\neg Q}\gg\wedge_1$$

and $\mathcal{I}^-$ ends in $\Gamma;\mathcal{D}\vdash D\gg Q$. By IH 2.3

Case:

$$\mathcal{I}^- = \frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not_D}(D'_2)\gg\neg Q}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not_D}(D'_1\vee D'_2)\gg\neg Q}\gg\wedge_2$$

and $\mathcal{I}^-$ ends in $\Gamma;\mathcal{D}\vdash D\gg Q$. By IH 2.3.

Case:

$$\mathcal{I}^- = \frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not_D}(D'_1)\gg\neg Q \qquad \Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not_D}(D'_2)\gg\neg Q}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not_D}(D'_1\wedge D'_2)\gg\neg Q}\gg\vee$$

and $\mathcal{I}^-$ ends in $\Gamma;\mathcal{D}\vdash D\gg Q$. By IH 2.3.

$\boxed{2.4}$ Let $D\sqsubseteq \mathrm{def}(q,\mathcal{D})$ and $D'\sqsubseteq \mathrm{def}(q,\mathcal{D})$. There are two main cases, according to whether D and D' belong to the same block (and in this case it suffices to consider $D\equiv D'$) or not:

1. It is not the case that $\mathcal{I}^+ :: \Gamma;\mathcal{D}\vdash_{\mathcal{P}} D\gg Q$ and $\mathcal{I}^- :: \Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D)\gg\neg Q$. By mutual induction on the structure of $\mathcal{I}^+$ and $\mathcal{S}^+$.

Case:

$$\mathcal{I}^+ = \frac{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} [\sigma]\overline{N_n} \doteq \overline{M_n} \qquad \Gamma;\mathcal{D}\vdash_{\mathcal{P}} [\sigma]G}{\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \forall(q\ \overline{N_n}\leftarrow G)\gg q\ \overline{M_n}}\gg\rightarrow$$

and $\mathcal{I}^-$ ends in $\Gamma;\mathcal{D}\vdash_{\mathcal{P}} \mathrm{Not}_\alpha(\forall(q\ \overline{N_n}\leftarrow G))\gg\neg q\ \overline{M_n}$.

$\mathcal{S}_1^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G$ By sub-derivation

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_n} \doteq \overline{M_n}$ By sub-derivation

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{x \in \mathrm{dom}(\Gamma)} \mathrm{Not}_x^C(q\ \overline{N_n}) \wedge (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}_x^i(q\ \overline{N_n}))) \wedge \forall(\neg q\ \overline{N_n}) \leftarrow \mathrm{Not}_{\mathrm{G}}(G) \gg \neg q\ \overline{M_n}$
By rule $\mathrm{Not}_\alpha \rightarrow$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]N_i \doteq M_i$ (+) for all $1 \le i \le n$ By inversion

$\Gamma \vdash M_i \in \|N_i\|$ (*) for all $1 \le i \le n$ By inversion

Subcase: $[\sigma]N_i \not\equiv e_x$, $[\sigma]N_i$ rigid:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_x^C(q\ \overline{N_n}) \gg \neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le i \le n} \forall Z_1 : A_1. \ldots \forall Z_n : A_n. \neg q\ [e_x/Z_j]\overline{Z_n} \gg \neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} e_x \doteq M_j$ By inversion

$\perp$ Above line and (+).

Subcase: $[\sigma]N_i \equiv e_x$:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}_x^i(q\ \overline{N_n}))) \gg [\sigma]\neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_x^i(\overline{N_n}) \gg \neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le i \le n} \bigwedge_{N \in \Gamma \vdash \mathrm{Not}(M_i)} \forall(\neg q\ [e_x/Z_i, N/Z_j]\overline{Z_n} \leftarrow \top) \gg \neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{N \in \Gamma \vdash \mathrm{Not}(M_i)} \forall(\neg q\ [e_x/Z_i, N/Z_j]\overline{Z_n} \leftarrow \top) \gg \neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} M_i \doteq [\theta]N$ By inversion

$\Gamma \vdash M_i \in \|N\|$ for some $N \in \Gamma \vdash \mathrm{Not}(N_i)$ By definition of $\|\_\|$

$\perp$ By term exclusivity (4.21) with line (*)

Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \neg\forall(q\ \overline{N_n} \leftarrow \mathrm{Not}_{\mathrm{G}}(G)) \gg \neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}([\sigma]G)$ By inversion

$\perp$ By IH 1 on $\mathcal{S}_1^+$

Case: $\mathcal{I}^+$ ends in $\gg\vee, \gg\wedge, \gg\forall$: by IH 2.4.1 similarly to case 2.1 with Not_α in place of $\mathrm{Not}_{\mathrm{D}}$.

2. It is not the case that $\mathcal{I}^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ and $\mathcal{I}^- :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D') \gg \neg Q$. By simultaneous induction on the structure of $\mathcal{I}^+$ and $\mathcal{I}^-$:

No Case: $\mathcal{I}^+$ ends in $\gg\top$.

No Case: $\mathcal{I}^+$ ends in $\gg\perp$, by restriction to complementable clauses and schema satisfaction.

No Case: $\mathcal{I}^+$ ends in $\gg\rightarrow$, $\mathcal{I}^-$ ends in $\gg\perp$, as above.

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_i^y} \doteq \overline{M_n} \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(q\ \overline{N_i^y} \leftarrow G) \gg q\ \overline{M_n}} \gg\rightarrow$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(\forall(q\ \overline{S_n} \leftarrow G')) \gg \neg q\ \overline{M_n}$, from a different block.

$\mathcal{S}_1^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G$ By sub-derivation

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_i^y} \doteq \overline{M_n}$ By sub-derivation

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{x \in \mathrm{dom}(\Gamma)} \mathrm{Not}_x^C(q\ \overline{S_n}) \wedge (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}_x^i(q\ \overline{S_n}))) \wedge \forall(\neg q\ \overline{S_n}) \leftarrow \mathrm{Not}_{\mathrm{G}}(G') \gg \neg q\ \overline{M_n}$
By rule $\mathrm{Not}_\alpha \rightarrow$, $y \not\equiv x$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]N_i \doteq M_i$ (+) for all $1 \le i \le n$ By inversion

Subcase: $[\sigma]N_i \not\equiv e_{yy}$, $[\sigma]N_i$ rigid:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_x^C(q\ \overline{S_n}) \gg \neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le i \le n} \forall Z_1 : A_1. \ldots \forall Z_n : A_n. \neg q\ [e_x/Z_j]\overline{Z_n} \gg \neg q\ \overline{M_n}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} e_x \doteq M_j$ By inversion

$\perp$ Above line, N_j rigid and (+).

Subcase: $[\sigma]N_i \equiv_y$:

$\perp$ Above line, eigenvariable condition and (+).

Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \neg\forall(q\ \overline{S_n} \leftarrow \mathrm{Not}_{\mathrm{G}}(G')) \gg \neg q\ \overline{M_n}$ By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{S_n} \doteq \overline{M_n}$ By inversion
$\bot$ Above line, eigenvariable condition and (+).

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_n} \doteq \overline{M_n} \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(q\ \overline{N_n} \leftarrow G) \gg q\ \overline{M_n}} \gg \rightarrow$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\alpha}(\top_q)) \gg \neg q\ \overline{M_n}$, from a different block.
$\mathcal{S}_1^+ :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]G$ By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_n} \doteq \overline{M_n}$ By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{x \in \mathrm{dom}(\Gamma)} (\bigwedge_{1 \le i \le n, x R^i q} \mathrm{Not}_x^C(\top_q)) \gg \neg q\ \overline{M_n}$
By rule $\mathrm{Not}_{\alpha}\top$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]N_i \doteq M_i$ (+) for all $1 \le i \le n$ By inversion
$\Gamma \vdash M_i \in \|N_i\|$ (∗) for all $1 \le i \le n$ By inversion
$[\sigma]N_i \not\equiv e_x$, $[\sigma]N_i$ rigid: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_x^C(q\ \overline{N_n}) \gg \neg q\ \overline{M_n}$ By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le i \le n} \forall Z_1{:}A_1. \ldots \forall Z_n{:}A_n. \neg q\ [e_x/Z_j]\overline{Z_n} \gg \neg q\ \overline{M_n}$ By inversion
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} e_x \doteq M_j$ By inversion
$\bot$ Above line and (+).

Case:

$$\mathcal{I}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (D_1 \wedge D_2) \gg Q} \gg\wedge_1$$

and $\mathcal{I}^-$ ends in $\Gamma; \mathcal{D} \vdash \mathrm{Not}_{\alpha}(D') \gg \neg Q$:
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D_1 \gg Q$ By sub-derivation
$\bot$ By IH 2.4.2

Case: $\mathcal{I}^+$ ends in $\gg\wedge_1, \gg\vee$, $\mathcal{I}^-$ ends in $\gg\wedge_i, \gg\vee$: analogously by IH 2.4.2.

$\boxed{1}$ By induction on the structure of $\mathcal{S}^+$.

Case:

$$\mathcal{S}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{def}(q, \mathcal{P} \wedge \mathcal{D}) \gg Q}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} Q} \vdash \mathrm{atm}$$

and $\mathcal{S}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(Q)$:
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{def}(q, \mathcal{P} \wedge \mathcal{D}) \gg Q$ By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \neg Q$ By rule $\mathrm{Not}_{\mathrm{G}}\mathrm{At}$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{def}(\neg q, \mathcal{P} \wedge \mathcal{D}) \gg \neg Q$ By inversion

Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathit{mnotd}(\mathrm{def}(q, \mathcal{P}))$ By definition
$\bot$ By IH 2.i, $i \in \{1, 3\}$ according to D static or dynamic
Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathit{mnota}(\mathrm{def}(q, \mathcal{D}))$ By definition
$\bot$ By IH 2.i, $i \in \{2, 4\}$ according to D static or dynamic

Case:

$$\mathcal{S}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1 \qquad \Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1 \wedge G_2} \vdash\wedge$$

and $\mathcal{S}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G_1 \wedge G_2)$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$ By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2$ By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G_1) \vee \mathrm{Not}_{\mathrm{G}}(G_2)$ By rule $\mathrm{Not}_{\mathrm{G}}\wedge$

Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G_1)$ By inversion
$\bot$ By IH 1

Subcase: $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G_2)$ By inversion
$\perp$ By IH 1

Case:

$$\mathcal{S}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1 \vee G_2} \vdash \vee$$

and $\mathcal{S}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G_1 \vee G_2)$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1$ By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G_1) \wedge \mathrm{Not}_{\mathrm{G}}(G_2)$ By rule $\mathrm{Not}_{\mathrm{G}}\vee$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G_1)$ By inversion
$\perp$ By IH 1

Case:

$$\mathcal{S}^+ = \frac{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_2}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} G_1 \vee G_2} \vdash \vee$$

and $\mathcal{S}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G_1 \vee G_2)$. Symmetrical.

Case:

$$\mathcal{S}^+ = \frac{(\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} [y/x]G'}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall x : a.\, G'} \vdash\forall^*$$

and $\mathcal{S}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(\forall x : a.\, G')$:

$(\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} [y/x]G'$ By sub-derivation
$(\Gamma, y{:}A); \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}([y/x]G')$ By rule $\mathrm{Not}_{\mathrm{G}}\forall$
$\perp$ By IH 1

Case:

$$\mathcal{S}^+ = \frac{\Gamma; (\mathcal{D} \wedge D') \vdash_{\mathcal{P}} G'}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D' \to G'} \vdash \to$$

and $\mathcal{S}^-$ ends in $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(D' \to G')$:

$\Gamma; (\mathcal{D} \wedge D') \vdash_{\mathcal{P}} G'$ By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D' \to \mathrm{Not}_{\mathrm{G}}(G')$ By rule $\mathrm{Not}_{\mathrm{G}} \to$
$\Gamma; (\mathcal{D} \wedge D') \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(G')$ By inversion
$\perp$ By IH 1

Case: The (dis)equality case follows from the decidability of the (dis)equality judgment.

□

Note that the proof goes through as there is no 'bad' interaction between the static and dynamic definition of a predicate; namely in sub-case 2.2 there is no overlap between a clause from $\mathrm{def}(q, \mathcal{P})$ and $\mathrm{def}(\neg q, \mathcal{D})$ since in every atomic assumption there must be an occurrence of a new parameter *and* every term at the same position in a program clause head must be *rigid*. Symmetrically for 2.3. Sub-case 2.4 holds analogously to the first case 2.1, which is based on term exclusivity (Corollary 4.21). In the latter case, it suffices to consider, for $D \sqsubseteq \mathrm{def}(q, \mathcal{P})$, only $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D \gg Q$ and $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{D}}(D) \gg \neg Q$, because the positive definition is a conjunction, while the negative one is a disjunction; 2.4 needs to discriminate whether the two dynamic assumptions belong to the same block: if so the same remark applies. If not, the (pivot) eigenvariable condition ensures non-overlapping.

The condition for programs to require clause heads with rigid terms may seem too restrictive; the natural way to relax it, requiring that only *one* of those terms which occur in the same position of a parameter in each assumption needs to be rigid, turns out to be inconsistent; in fact, the property needs to be preserved in the negative program as well. Consider the following counterexample:

$$\begin{array}{lcl} plam & : & p\ (lam\ E)\ (lam\ F) \leftarrow \forall x : exp.\, p\ (E\ x)\ (F\ x). \\ papp & : & p\ (app\ E_1\ E_2)\ F. \end{array}$$

Augmentation will insert the clause $\forall(\neg p\ x\ H \wedge \neg p\ H'\ x)$, which now overlaps with $\neg\mathtt{papp}$. As a result both $p\ (lam(\lambda x\!:\!exp.\, app\ x\ x))\ (lam(\lambda y\!:\!exp.\, y))$ and its negation are provable.

As we remarked earlier, the rigidity condition applies only to predicate definitions which are mutually recursive to non-Horn ones. Thus, any such program which uses a 'catch-all' clause is those positions is forbidden; nevertheless it is easy to avoid catch-all clauses via explicit coercion or (sometimes) partial evaluation. Finally, we remark that any other (decidable) condition which avoids overlap between static and dynamic clauses will do; the one we have proposed is statically checkable, certainly not 'ad hoc' and has been dictated by the practice of logical frameworks.

1.8 Exhaustivity

Theorem 1.35 (Exhaustivity) *Let* $\models_{\mathcal{S}^a} aug_D(\mathcal{P})$ *and* $\Gamma;\mathcal{D} < \mathcal{S}^a$. *For every goal* G *such that* $\Gamma;\mathcal{D}\backslash G < \mathcal{S}^a$, *if* $\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G$ *then* $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G)$.

Proof: Note that $\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{def}(\neg q, \mathcal{P} \wedge \mathcal{D}) \gg \neg Q$ iff so does every disjunct. We generalize this to:

1. If $\mathcal{S}^- :: \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} G$, then $\mathcal{S}^+ :: \Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G)$.

2.1 If $D \sqsubseteq \mathrm{def}(Q,\mathcal{P})$ $\mathcal{I}^- :: \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D \gg Q$, then $\mathcal{I}^+ :: \Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D) \gg \neg Q$.

2.2 If $D \sqsubseteq \mathrm{def}(Q,\mathcal{D})$ $\mathcal{I}^- :: \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} D \gg Q$, then $\mathcal{I}^+ :: \Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\alpha}(D) \gg \neg Q$.

The proof is by mutual induction on the structure of $\mathcal{S}^-$ and $\mathcal{I}^-$. We start with part 2.1:

$\boxed{2.1}$

Case:

$$\mathcal{I}^- = \frac{}{\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} \top \gg Q}\ \not\gg\top$$

Then $\mathrm{Not_D}(\top) = \bot$ and

$$\mathcal{I}^+ = \frac{}{\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \bot \gg \neg Q}\ \gg\bot$$

Case:

$$\mathcal{I}^- = \frac{\text{for all } \theta\ \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} [\theta]\overline{N_n} \doteq \overline{M_n}}{\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} \forall(q\ \overline{N_n} \leftarrow G) \gg q\ \overline{M_n}}\ \not\gg\to^1$$

for all θ $\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} [\theta]\overline{N_n} \doteq \overline{M_n}$	By sub-derivation
for all θ $[\theta]\overline{N_n} \neq \overline{M_n}$	By rule $\nvdash\doteq$
$\cdot \vdash \overline{M_n} \notin \lVert\overline{N_n}\rVert$	By definition of $\lVert\_\rVert$
$\cdot \vdash \overline{M_n} \in \lVert\mathrm{Not}(\overline{N_n})\rVert$	By term exhaustivity (4.21)
$\overline{M_n} = [\sigma]\overline{S_n}$ for some σ and $\overline{S_n} \in \cdot \vdash \mathrm{Not}(\overline{N_n})$	By definition of $\lVert\_\rVert$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \overline{M_n} \doteq [\sigma]\overline{S_n}$	By rule $\vdash\doteq$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \forall(\neg q\ \overline{S_n} \leftarrow \top) \gg \neg q\ \overline{M_n}$	By rule $\gg\forall$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{\overline{S_n}\in\cdot\vdash\mathrm{Not}(\overline{N_n})} \forall(\neg q\ \overline{S_n} \leftarrow \top) \gg \neg q\ \overline{M_n}$	By appropriate applications of rule $\gg\wedge$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{\overline{S_n}\in\cdot\vdash\mathrm{Not}(\overline{N_n})} \forall(\neg q\ \overline{S_n} \leftarrow \top) \wedge \forall(\neg q\ \overline{M_n} \leftarrow \mathrm{Not_G}(G)) \gg \neg q\ \overline{M_n}$	By rule $\gg\wedge_1$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(\forall(q\ \overline{N_n} \leftarrow G) \gg \neg q\ \overline{M_n})$	By rule $\mathrm{Not_D}\gg$

Case:

$$\mathcal{I}^- = \frac{\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} [\sigma]\overline{N_n} \not\doteq \overline{M_n} \qquad \Gamma;\mathcal{D} \nvdash_{\mathcal{P}} [\sigma]G}{\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} \forall(q\ \overline{N_n} \leftarrow G) \gg q\ \overline{M_n}}\ \not\gg\to_2$$

$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} [\sigma]\overline{N_n} \not\doteq \overline{M_n}$	By sub-derivation
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_n} \doteq \overline{M_n}$	By rule $\vdash\not\doteq$
$\Gamma;\mathcal{D} \nvdash_{\mathcal{P}} [\sigma]G$	By sub-derivation
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}[\sigma](G)$	By IH 1
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \forall(\neg q\ \overline{N_n} \leftarrow \mathrm{Not_G}(G)) \gg \neg q\ \overline{M_n}$	By rule $\gg\to$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{\overline{S_n}\in\cdot\vdash\mathrm{Not}(\overline{N_n})} \forall(\neg q\ \overline{S_n} \leftarrow \top) \wedge \forall(\neg q\ \overline{N_n} \leftarrow \mathrm{Not_G}(G)) \gg \neg q\ \overline{M_n}$	By rule $\gg\wedge_2$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(\forall(q\ \overline{N_n} \leftarrow G)) \gg \neg q\ \overline{M_n}$	By rule $\mathrm{Not_D}\to$

Case:

$$\mathcal{I}^- = \frac{\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q}{\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} (D_1 \vee D_2) \gg Q} \not\gg\vee_1$$

$\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q$ By sub-derivation
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1) \gg \neg Q$ By IH 2.1
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1) \wedge \mathrm{Not_D}(D_2) \gg \neg Q$ By rule $\gg\wedge_1$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1 \vee D_2) \gg \neg Q$ By rule $\mathrm{Not_D}\vee$

Case:

$$\mathcal{I}^- = \frac{\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} (D_1 \vee D_2) \gg Q} \not\gg\vee_2$$

Symmetrical to the above.

Case:

$$\mathcal{I}^- = \frac{\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q \qquad \Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} (D_1 \wedge D_2) \gg Q} \not\gg\wedge$$

$\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q$ By sub-derivation
$\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} D_2 \gg Q$ By sub-derivation
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1) \gg \neg Q$ By IH 2.1
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_2) \gg \neg Q$ By IH 2.1
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1) \vee \mathrm{Not_D}(D_2) \gg \neg Q$ By rule $\gg\vee$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D_1 \wedge D_2) \gg \neg Q$ By rule $\mathrm{Not_D}\wedge$

$\boxed{2.2}$ Recall that $q\ \overline{M^i_{e_x}}$ stands for $q\ M_1 \ldots M_{i-1}\ e_x\ M_{i+1} \ldots M_n$, for $x{:}A \in \Gamma$:

Case:

$$\mathcal{I}^- = \frac{}{\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} \top_q \gg q\ \overline{M^i_{e_x}}} \not\gg\top$$

$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} [\overline{M^i_{e_x}}/\overline{Z_n}]\overline{Z_n} \doteq \overline{M^i_{e_x}}$ By rule $\vdash\doteq$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \forall Z_1{:}A_1.\ldots\forall Z_n{:}A_n.\neg q\ [e_x/Z_i]\overline{Z_n\,Z_n} \gg \neg q\ \overline{M^i_{e_x}}$ By rule $\gg\forall$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}^C_x(\top_q) \gg \neg q\ \overline{M^i_{e_x}}$ By rule Not^C_x
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{x \in dom(\Gamma)} \bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}^i_x(\top_q) \gg \neg q\ \overline{M^i_{e_x}}$ By appropriate applications of rule $\gg\wedge$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(\top_q) \gg \neg q\ \overline{M^i_{e_x}}$ By rule $\mathrm{Not}_\alpha\top$

Case:

$$\mathcal{I}^- = \frac{x{:}a \in \Gamma,\ \text{for all } \theta\ \Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} \overline{N_n} \doteq \overline{M^i_{e_x}}}{\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} \forall(q\ \overline{N_n} \leftarrow G) \gg q\ \overline{M^i_{e_x}}} \not\gg\rightarrow^1$$

for all θ $\Gamma;\mathcal{D} \not\vdash_{\mathcal{P}} [\theta]\overline{N_n} \doteq q\ \overline{M^i_{e_x}}$ and $x{:}a \in \Gamma$ By sub-derivation

Subcase: $e_x \neq [\theta]N_i$, N_i rigid by definition for some $1 \le i \le n$:

$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} [\overline{M^i_{e_x}}/\overline{Z_n}]\overline{Z_n} \doteq \overline{M^i_{e_x}}$ By rule $\vdash\doteq$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \forall Z_1{:}A_1.\ldots\forall Z_n{:}A_n.\neg q\ [e_x/Z_i]\overline{Z_n} \gg \neg q\ \overline{M^i_{e_x}}$ By rule $\gg\forall$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{1 \le i \le n} \forall Z_1{:}A_1.\ldots\forall Z_n{:}A_n.\neg q\ [e_x/Z_i]\overline{Z_n}) \gg \neg q\ \overline{M^i_{e_x}}$
By appropriate applications of rule $\gg\wedge$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}^C_x(q\ \overline{N_n}) \gg \neg q\ \overline{M^i_{e_x}}$ By rule Not^C_x
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{x \in dom(\Gamma)} \mathrm{Not}^C_x(q\ \overline{M_n}) \wedge (\bigwedge_{1 \le i \le n, xR^iq} \Gamma \vdash \mathrm{Not}^i_x(q\ \overline{M_n}))) \wedge \forall(\neg q\ \overline{M_n}) \leftarrow G' \gg \neg q\ \overline{M^i_{e_x}}$
By appropriate applications of rule $\gg\wedge$
$\Gamma;\mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(\forall(q\ \overline{N_n} \leftarrow G)) \gg \neg q\ \overline{M^i_{e_x}}$ By rule $\mathrm{Not}_\alpha \rightarrow$

Subcase: $e_x \doteq [\theta]N_i, M_j \neq [\theta]N_j$ for $1 \le i, j \le n, i \neq i$:

for all θ $[\theta]N_j \neq M_j$ By rule $\not\vdash\doteq$
$\Gamma \vdash M_j \notin \|N_j\|$ By definition of $\|\_\|$
$\Gamma \vdash M_j \in \|\mathrm{Not}(N_j)\|$ By term exhaustivity (4.21)

$M_j = [\sigma]N$ for some σ and $N \in \Gamma \vdash \mathrm{Not}(N_j)$ By definition of $\|\_\|$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} M_j \doteq [\sigma]N$ By rule $\vdash\doteq$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(\neg q\ [e_x/Z_i, N/Z_j]\overline{Z_n} \leftarrow \top) \gg \neg q\ \overline{M^i_{e_x}}$ By rule $\gg\rightarrow$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{N \in \Gamma \vdash \mathrm{Not}(M_j)} \forall(\neg q\ [e_x/Z_i, N/Z_j]\overline{Z_n} \leftarrow \top) \gg \neg q\ \overline{M^i_{e_x}}$

By appropriate applications of rule $\gg\wedge$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \bigwedge_{1 \le j \le n} \bigwedge_{N \in \Gamma \vdash \mathrm{Not}(M_j)} \forall(\neg q\ [e_x/Z_i, N/Z_j]\overline{Z_n} \leftarrow \top) \gg \neg q\ \overline{M^i_{e_x}}$

By appropriate applications of rule $\gg\wedge$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}^i_x(\overline{N_n}) \gg \neg q\ \overline{M^i_{e_x}}$ By rule Not^i_x

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}^i_x(\overline{N_n}) \gg \neg q\ \overline{M^i_{e_x}}$

By appropriate applications of rule $\gg\wedge$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{x \in dom(\Gamma)} \mathrm{Not}^C_x(q\ \overline{M_n}) \wedge (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}^i_x(q\ \overline{M_n}))) \gg \neg q\ \overline{M^i_{e_x}}$

By appropriate applications of rule $\gg\wedge$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{x \in dom(\Gamma)} \mathrm{Not}^C_x(q\ \overline{M_n}) \wedge (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}^i_x(q\ \overline{M_n}))) \wedge \forall(\neg q\ \overline{M_n} \leftarrow G') \gg \neg q\ \overline{M^i_{e_x}}$

By rule $\gg\wedge_2$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(\forall(q\ \overline{N_n} \leftarrow G)) \gg \neg q\ \overline{M^i_{e_x}}$ By rule $\mathrm{Not}_\alpha\rightarrow$

Case:

$$\mathcal{I}^- = \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [\sigma]\overline{N_n} \not\doteq \overline{M^i_{e_x}} \qquad \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [\sigma]G}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \forall(q\ \overline{N_n} \leftarrow G) \gg q\ \overline{M^i_{e_x}}} \not\gg\rightarrow_2$$

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [\sigma]\overline{N_n} \not\doteq \overline{M^i_{e_x}}$ By sub-derivation

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\sigma]\overline{N_n} \doteq \overline{M^i_{e_x}}$ By rule $\vdash\not\doteq$

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} [\sigma]G$ By sub-derivation

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}[\sigma](G)$ By IH 1

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \forall(\neg q\ \overline{N_n} \leftarrow \mathrm{Not}_{\mathrm{G}}(G)) \gg \neg q\ \overline{M^i_{e_x}}$ By rule $\gg\rightarrow$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} (\bigwedge_{x \in dom(\Gamma)} \mathrm{Not}^C_x(q\ \overline{M_n}) \wedge (\bigwedge_{1 \le i \le n, xR^iq} \mathrm{Not}^i_x(q\ \overline{M_n}))) \wedge \forall(\neg q\ \overline{M_n}) \leftarrow G' \gg \neg q\ \overline{M^i_{e_x}}$

By rule $\gg\wedge_2$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(\forall(q\ \overline{N_n} \leftarrow G)) \gg \neg q\ \overline{M^i_{e_x}}$ By rule $\mathrm{Not}_\alpha\rightarrow$

Case:

$$\mathcal{I}^- = \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} (D_1 \vee D_2) \gg Q} \not\gg\vee_1$$

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q$ By sub-derivation

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D_1) \gg \neg Q$ By IH 2.2

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D_1) \wedge \mathrm{Not}_\alpha(D_2) \gg \neg Q$ By rule $\gg\wedge_1$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D_1 \vee D_2) \gg \neg Q$ By rule $\mathrm{Not}_\alpha\vee$

Case:

$$\mathcal{I}^- = \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} (D_1 \vee D_2) \gg Q} \not\gg\vee_2$$

Symmetrical to the above

Case:

$$\mathcal{I}^- = \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q \qquad \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_2 \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} (D_1 \wedge D_2) \gg Q} \not\gg\wedge$$

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_1 \gg Q$ By sub-derivation

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D_2 \gg Q$ By sub-derivation

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D_1) \gg \neg Q$ By IH 2.2

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D_2) \gg \neg Q$ By IH 2.2

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D_1) \vee \mathrm{Not}_\alpha(D_2) \gg \neg Q$ By rule $\gg\vee$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_\alpha(D_1 \wedge D_2) \gg \neg Q$ By rule $\mathrm{Not}_{\mathrm{D}}\wedge$

$\boxed{1}$

Case:

$$\mathcal{S}^- = \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \mathrm{def}(Q, \mathcal{P} \wedge \mathcal{D}) \gg Q}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} Q} \not\vdash \mathrm{atm}$$

Subcase: $D \sqsubseteq \mathrm{def}(Q, \mathcal{P})$

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \gg Q$ — By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_D}(D) \gg \neg Q$ — By IH 2.1
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{def}(\neg Q, \mathcal{P}) \gg \neg Q$ — By repeated applications of rule $\gg\vee$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(Q)$ — By rule $\vdash \mathrm{At}$

Subcase: $D \sqsubseteq \mathrm{def}(Q, \mathcal{D})$

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D \gg Q$ — By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\alpha}(D) \gg \neg Q$ — By IH 2.2
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{def}(\neg Q, \mathcal{D}) \gg \neg Q$ — By repeated applications of $\gg\vee$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(Q)$ — By rule $\vdash \mathrm{At}$

Case:

$$\mathcal{S}^- = \frac{}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \bot} \not\vdash \bot$$

Then $\mathrm{Not_G}(\bot) = \top = \top$ and

$$\mathcal{S}^+ = \frac{}{\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(\top)} \vdash \top$$

Case:

$$\mathcal{S}^- = \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_1 \qquad \Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_2}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} (G_1 \vee G_2)} \not\vdash \vee$$

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_1$ — By sub-derivation
$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_2$ — By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G_1)$ — By IH 1
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G_2)$ — By IH 1
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G_1) \wedge \mathrm{Not_G}(G_2)$ — By rule $\vdash \wedge$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G_1 \vee G_2)$ — By rule $\mathrm{Not_G}\wedge$

Case:

$$\mathcal{S}^- = \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_1}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} (G_1 \wedge G_2)} \not\vdash \wedge_1$$

$\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_1$ — By sub-derivation
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G_1)$ — By IH 1
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G_1) \vee \mathrm{Not_G}(G_2)$ — By rule $\vdash \vee$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(G_1 \wedge G_2)$ — By rule $\mathrm{Not_G}\vee$

Case:

$$\mathcal{S}^- = \frac{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} G_2}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} (G_1 \wedge G_2)} \not\vdash \wedge_2$$

Symmetrical to the above.

Case:

$$\mathcal{S}^- = \frac{\Gamma; \mathcal{D} \wedge D' \not\vdash_{\mathcal{P}} G'}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} D' \to G'} \not\vdash \to$$

$\Gamma; \mathcal{D} \wedge D' \not\vdash_{\mathcal{P}} G'$ — By sub-derivation
$\Gamma; \mathcal{D} \wedge D' \vdash_{\mathcal{P}} \mathrm{Not_G}(G')$ — By IH 1
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} D' \to \mathrm{Not_G}(G')$ — By rule $\vdash\to$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not_G}(D' \to G')$ — By rule $\mathrm{Not_G} \to$

Case:

$$\mathcal{S}^- = \frac{(\Gamma, y{:}a); \mathcal{D} \not\vdash_{\mathcal{P}} [y/x]G'}{\Gamma; \mathcal{D} \not\vdash_{\mathcal{P}} \forall x : a.\, G'} \not\vdash \forall$$

$(\Gamma, y{:}a); \mathcal{D} \not\vdash_{\mathcal{P}} [y/x]G'$	By sub-derivation
$(\Gamma, y{:}a); \mathcal{D} \vdash_{\mathcal{P}} [y/x]\mathrm{Not}_{\mathrm{G}}(G')$	By IH 1
$(\Gamma, y{:}a); \mathcal{D} \vdash_{\mathcal{P}} \forall x : a.\, \mathrm{Not}_{\mathrm{G}}(G')$	By rule $\vdash \forall$
$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} \mathrm{Not}_{\mathrm{G}}(\forall x : a.\, G')$	By rule $\mathrm{Not}_{\mathrm{G}}\forall$

Case: (dis)equations: immediate.

□

Corollary 1.36 *Clause complementation satisfies the boolean rules of negation.*

1.9 Refinements

In the following sections we will move towards an operational semantics for our language.

1.9.1 More on Termination

We prove that clause complementation preserve termination, so that a negative program can be used as a decision procedure if so is its positive counterpart. First some preliminaries: define $\mathrm{Not}(R)$ as follows:

$$\begin{aligned} \mathrm{Not}(\cdot) &= \cdot \\ \mathrm{Not}(R, G_1 \prec G_2) &= \mathrm{Not}(R), \mathrm{Not}_{\mathrm{G}}(G_1) \prec \mathrm{Not}_{\mathrm{G}}(G_2) \end{aligned}$$

Observe that, by the restriction to clauses of the form $\forall(Q \leftarrow G)$, it is sufficient to consider only goal complementation. It is clear that negation is preserved under relation union, that is:

$$\mathrm{Not}(R_1, R_2) = \mathrm{Not}(R_1), \mathrm{Not}(R_2)$$

We will also need the following technical Lemma, stating that goal complementation commutes with substitution:

Lemma 1.37 *Let* $\Gamma \vdash \theta : \Phi$. $\Gamma \vdash_{\Phi} \mathrm{Not}_{\mathrm{G}}([\theta]G) = \Gamma \vdash_{\Phi} [\theta]\mathrm{Not}_{\mathrm{G}}(G)$.

Proof: By a simple induction on the structure of the derivation of $\Gamma \vdash_{\Phi} \mathrm{Not}_{\mathrm{G}}(G)$. □

Lemma 1.38 *If* $[R]$ *is well-founded, so is* $[\mathrm{Not}(R)]$.

Proof: Suppose $[\mathrm{Not}(R])$ is not well-founded. Then there is an infinite descending chain $\ldots \prec [\theta]G_i \prec \ldots \prec [\theta]G_2 \prec [\theta]G_1$. By definition of $\mathrm{Not}(R)$ there is G'_i such that $\mathrm{Not}_{\mathrm{G}}(G'_i) = G_i$ and that $G'_i \prec G'_{i-1} \in R$ for some $i \in \omega$. Since by Lemma ?? $\mathrm{Not}_{\mathrm{G}}([\theta]G_i) = [\theta]\mathrm{Not}_{\mathrm{G}}(G_i)$, then $[R]$ allows an infinite descending chain $\ldots \prec [\theta]G'_i \prec \ldots \prec [\theta]G'_2 \prec [\theta]G'_1$, impossible. □

The final piece is to show that the relation induced by the complement of a program is the complement of the relation induced by the program itself.

Lemma 1.39

1. *If* $\pi :: D \overset{D}{\Longrightarrow} R$ *then* $\mathrm{Not}_{\mathrm{D}}(D) \overset{D}{\Longrightarrow} \mathrm{Not}(R)$.
2. *If* $\alpha :: D \overset{D}{\Longrightarrow} R$ *then* $\mathrm{Not}_{\alpha}(D) \overset{D}{\Longrightarrow} \mathrm{Not}(R)$.
3. *If* $\gamma :: \Gamma \vdash G \overset{G}{\Longrightarrow} R$ *then* $\Gamma \vdash \mathrm{Not}_{\mathrm{G}}(G) \overset{G}{\Longrightarrow} \mathrm{Not}(R)$.

Proof: By induction on the structure of the given derivations. We prove some selected cases:

Case: $\pi :: \top \overset{D}{\Longrightarrow} \cdot$; $\mathrm{Not}_{\mathrm{D}}(\top) = \bot \overset{D}{\Longrightarrow} \cdot = \mathrm{Not}(\cdot)$.

Case: π ends in $\forall(q\ \overline{M_n} \leftarrow G) \stackrel{D}{\Longrightarrow} R, [\rho]G \prec [\rho]q\ \overline{M_n}$, for a global substitution ρ:

$\cdot \vdash [\rho]G \stackrel{G}{\Longrightarrow} R$ — By sub-derivation

$\cdot \vdash \mathrm{Not_G}([\rho]G) \stackrel{G}{\Longrightarrow} \mathrm{Not}(R)$ — By IH 2

$\forall(\neg q\ \overline{M_n} \leftarrow \mathrm{Not_G}(G)) \stackrel{D}{\Longrightarrow} \mathrm{Not}(R), \mathrm{Not}(G) \prec \neg q\ \overline{M_n}$ — By rule $\stackrel{D}{\Longrightarrow}\to$

$\bigwedge_{\overline{N_n} \in \mathrm{Not}(\overline{M_n})} \forall(\neg q\ \overline{N_n}) \stackrel{D}{\Longrightarrow} \cdot$ — By rule $\stackrel{D}{\Longrightarrow}\wedge$

$\bigwedge_{\overline{N_n} \in \mathrm{Not}(\overline{M_n})} \forall(\neg q\ \overline{N_n}) \wedge \forall(\neg q\ \overline{M_n} \leftarrow \mathrm{Not_G}(G)) \stackrel{D}{\Longrightarrow} \mathrm{Not}(R), \mathrm{Not}(G) \prec \neg q\ \overline{M_n}$ — By rule $\stackrel{D}{\Longrightarrow}\wedge$

$\bigwedge_{\overline{N_n} \in \mathrm{Not}(\overline{M_n})} \forall(\neg(q\ \overline{N_n})) \wedge \forall(\neg q\ \overline{M_n} \leftarrow \mathrm{Not_G}(G)) \stackrel{D}{\Longrightarrow} \mathrm{Not}(R), \mathrm{Not}(G) \prec \mathrm{Not_G}(q\ \overline{M_n})$ — By rule $\mathrm{Not_G}$At

$\mathrm{Not_D}(\forall(q\ \overline{M_n} \leftarrow G)) \stackrel{D}{\Longrightarrow} \mathrm{Not}(R, G \prec \neg q\ \overline{M_n})$ — By def. of $\mathrm{Not}(R)$

Case: $\gamma :: Q \stackrel{G}{\Longrightarrow} \cdot$; $\mathrm{Not_D}(Q) = \neg Q \stackrel{D}{\Longrightarrow} \cdot = \mathrm{Not}(\cdot)$.

Case: γ ends in $\Gamma \vdash G_1 \wedge G_2 \stackrel{G}{\Longrightarrow} R_1, R_2, G_1 \prec G_1 \wedge G_2, G_2 \prec G_1 \wedge G_2$:

$\Gamma \vdash G_1 \stackrel{G}{\Longrightarrow} R_1$ and $\Gamma \vdash G_2 \stackrel{G}{\Longrightarrow} R_2$ — By sub-derivation

$\Gamma \vdash \mathrm{Not_G}(G_1) \stackrel{G}{\Longrightarrow} \mathrm{Not}(R_1)$ and $\Gamma \vdash \mathrm{Not_G}(G_2) \stackrel{G}{\Longrightarrow} \mathrm{Not_G}(R_2)$ — By IH 2

$\Gamma \vdash \mathrm{Not}(G_1) \vee \mathrm{Not_G}(G_2) \stackrel{G}{\Longrightarrow} \mathrm{Not}(R_1), \mathrm{Not}(R_2),$
$\quad \mathrm{Not_G}(G_1) \prec \mathrm{Not_G}(G_1) \vee \mathrm{Not_G}(G_2), \mathrm{Not_G}(G_2) \prec \mathrm{Not_G}(G_1) \vee \mathrm{Not_G}(G_2)$ — By rule

$\Gamma \vdash \mathrm{Not}(G_1 \wedge G_2) \stackrel{G}{\Longrightarrow} \mathrm{Not}(R_1), \mathrm{Not}(R_2),$
$\quad \mathrm{Not_G}(G_1) \prec \mathrm{Not_G}(G_1 \wedge G_2), \mathrm{Not_G}(G_2) \prec \mathrm{Not}(G_1 \wedge G_2)$ — By rule

$\Gamma \vdash \mathrm{Not}(G_1 \wedge G_2) \stackrel{G}{\Longrightarrow} \mathrm{Not}(R_1, R_2, G_1 \prec G_1 \wedge G_2, G_2 \prec G_1 \wedge G_2)$ — By union and def. of $\mathrm{Not}(R)$

□

Corollary 1.40 *If $\mathcal{P}$ is terminating, so is* $\mathrm{Not_D}(\mathcal{P})$.

Proof: By definition, $\mathcal{P}\downarrow$ if $\mathcal{P} \stackrel{D}{\Longrightarrow} R$ and $[R]$ is well-founded: by Lemma ?? $\mathrm{Not_D}(\mathcal{P}) \stackrel{D}{\Longrightarrow} \mathrm{Not}(R)$ and by Lemma ?? $[\mathrm{Not}(R)]$ is well-founded. Thus $\mathrm{Not_D}(\mathcal{P})\downarrow$. □

1.9.2 Elimination of $\vee$

We now show how to eliminate the $\vee$ operator preserving provability; this will recover uniformity in proof-search. We generalize the approach in [?], where the operation was defined as follows: for $m_1 : Q_1 \leftarrow G_1$ and $m_2 : Q_2 \leftarrow G_2$:

$$m_1 \vee m_2 \quad = \quad \theta(Q_1 \leftarrow G_1 \wedge G_2) \quad \text{where } \theta = mgu(Q_1, Q_2)$$

As we remarked earlier, the $\vee$ operator was introduced to preserve the duality between conjunction and disjunction in clauses. Its use, as a clause constructor, is limited to clauses in the same predicate definition and therefore it can be eliminated simulating unification in the definition. Yet, the strict higher-order unification problem is quite complex and even more so complicated by the mixed quantifier structure of HHF. Moreover, we have already in our language variable-variable (dis)equations stemming from left-linearization and normalization of input variables. Finally, we have defined unification of simple terms in Section 4.3 as an intersection operation. This has greatly simplified the presentation, but does not immediately give us a notion of most general unifiers seen as sets of substitutions. We thus choose to compile our source into an intermediate language which makes unification problems explicit as simple equational problems in the style of the unification logic introduced in [?]. We can perform unification as constraint simplification as used in *Elf* [?] and *Twelf* [?] in a later stage that we do not describe here.

We adapt the *residuation* technique used in [?] to compile immediate implication into resolution. We define the judgment $D_1 \vee D_2 \setminus D$ in Figure ?? by simultaneous induction on D_1 and D_2, with the intended meaning of '$D_1 \vee D_2$ compiles to D'.

Example 1.41 *Continuing Example 6.26, recall that:*

$$\begin{aligned}
&\mathrm{Not_D}(cloz) = \\
&\qquad \forall F{:}exp.\neg closed\ (lam\ F) \wedge \forall F_1, F_2{:}exp.\neg closed\ (app\ F_1\ F_2). \\
&\mathrm{Not_D}(clolam) =
\end{aligned}$$

$$\frac{}{\top \vee D \setminus \top}\vee\top D \qquad \frac{}{D \vee \top \setminus \top}\vee D\top$$

$$\frac{}{\bot \vee D \setminus D}\vee\bot D \qquad \frac{}{D \vee \bot \setminus D}\vee D\bot$$

$$\frac{}{\forall(q\ \vec{N}_1 \leftarrow G_1) \vee \forall(q\ \vec{N}_2 \leftarrow G_2) \setminus \forall(q\ \vec{N}_1 \leftarrow (\vec{N}_1 \doteq \vec{N}_2) \wedge G_1 \wedge G_2)}\vee\mathrm{AtAt}$$

$$\frac{q \not\equiv q'}{\forall(q\ \vec{N}_1 \leftarrow G_1) \vee \forall(q'\ \vec{N}_2 \leftarrow G_2) \setminus \top}\vee\mathrm{AtAt} \not\equiv$$

$$\frac{D_1 \vee D_2 \setminus D \qquad D_1' \vee D_2 \setminus D'}{(D_1 \vee D_1') \vee D_2 \setminus D \vee D'}\vee\vee L \qquad \frac{D_1 \vee D_2 \setminus D \qquad D_1' \vee D_2 \setminus D'}{(D_1 \wedge D_1') \vee D_2 \setminus D \wedge D'}\vee\wedge L$$

$$\frac{D_1 \vee D_2 \setminus D \qquad D_1' \vee D_2 \setminus D'}{D_2 \vee (D_1 \vee D_1') \setminus D \vee D'}\vee\vee R \qquad \frac{D_1 \vee D_2 \setminus D \qquad D_1' \vee D_2 \setminus D'}{D_2 \vee (D_1 \wedge D_1') \setminus D \wedge D'}\vee\wedge R$$

Figure 1.14: $\vee$-Elimination: $D_1 \vee D_2 \setminus D$

$$\begin{array}{l}\neg closed\ z \wedge \forall F_1, F_2 : exp. \neg closed\ (app\ F_1\ F_2) \wedge \\ \forall E : exp. \neg closed\ (lam\ E) \leftarrow \forall x : exp.\ (closed\ x \rightarrow \neg closed\ (E\ x)).\end{array}$$

Removing trivially unsatisfiable clauses, the computation of $\mathrm{Not_D}(cloz) \vee \mathrm{Not_D}(clolam) \setminus D$ *results in:*

$$\begin{array}{l}\forall E : exp. \neg closed\ (lam\ E) \leftarrow (\forall x : exp.\ closed\ x \rightarrow \neg closed\ (E\ x)) \wedge \\ \forall F_1, F_2 : exp. \neg closed\ (app\ F_1\ F_2).\end{array}$$

Going back to the **linx** *example:*

$$\begin{array}{l}\mathrm{Not_D}(linxx) \vee \mathrm{Not_D}(linxap1) = \\ \quad (\neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x)) \wedge \neg linx(\lambda x\,.\,lam(\lambda y\,.\,(E\ x\ y)))) \vee \\ \quad (\neg linx(\lambda x\,.\,x) \wedge \neg linx(\lambda x\,.\,lam(\lambda y\,.\,(E\ x\ y))) \wedge \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x^1)) \\ \quad \wedge \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x)) \leftarrow \neg linx(\lambda x\,.\,E_1\ x))\end{array}$$

$\mathrm{Not_D}(linxx) \vee \mathrm{Not_D}(linxap1) \setminus D$, *where*

$$\begin{array}{l}D = \\ \quad \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x^1)) \wedge \\ \quad \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x)) \leftarrow \neg linx(\lambda x\,.\,E_1\ x) \wedge \\ \quad \neg linx(\lambda x\,.\,lam(\lambda y\,.\,E\ x\ y)).\end{array}$$

The following lemma guarantees that compilation preserves run-time immediate entailment:

Lemma 1.42 *Let* $D_1 \vee D_2 \setminus D$*: for every ground substitution* $\Gamma \vdash \theta : \Phi$*,* $\Gamma; \mathcal{D} \vdash [\theta](D_1 \vee D_2) \gg Q$ *iff* $\Gamma; \mathcal{D} \vdash [\theta]D \gg Q$*.*

Proof: $(\rightarrow)$. By induction on the structure of $\pi :: D_1 \vee D_2 \setminus D$.

Case: π ends in $\vee\top D, \vee\mathrm{AtAt} \not\equiv$: trivially true.

Case: π ends in $\vee\bot D, \vee\mathrm{At}\bot$: immediate.

Case:

$$\pi = \frac{}{\forall(q\ \vec{N}_1 \leftarrow G_1) \vee \forall(q\ \vec{N}_2 \leftarrow G_2) \setminus \forall(q\ \vec{N}_1 \leftarrow (\vec{N}_1 \doteq \vec{N}_2) \wedge G_1 \wedge G_2)}\vee\mathrm{atm}$$

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta](\forall(q\ \vec{N}_1 \leftarrow G_1) \vee \forall(q\ \vec{N}_2 \leftarrow G_2)) \gg q\ \vec{N}$ By hypothesis

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta](\forall(q\ \vec{N}_1 \leftarrow G_1)) \gg q\ \vec{N}$ and $[\theta](\forall(q\ \vec{N}_2 \leftarrow G_2)) \gg q\ \vec{N}$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta \cdot \sigma_1]\vec{N}_1 \doteq \vec{N}$ and $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta \cdot \sigma_1]G_1$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta \cdot \sigma_2]\vec{N}_2 \doteq \vec{N}$ and $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta \cdot \sigma_2]G_2$ By inversion

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta \cdot \sigma_1]\vec{N}_1 \doteq [\theta \cdot \sigma_2]\vec{N}_2$ By replacement

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta \cdot \sigma_1 \cdot \sigma_2]\vec{N}_1 \doteq [\theta \cdot \sigma_1 \cdot \sigma_2]\vec{N}_2 \wedge [\theta \cdot \sigma_1 \cdot \sigma_2]G_1 \wedge [\theta \cdot \sigma_1 \cdot \sigma_2]G_2$

By composition of ground substitution

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta](\forall(q\ \vec{N}_1 \leftarrow (\vec{N}_1 \doteq \vec{N}_2) \wedge G_1 \wedge G_2)) \gg q\ \vec{N}$ By rule

Case: π ends in $\vee$At$\wedge$, $\vee$At$\vee$, $\vee\wedge$, $\vee\vee$: by an immediate appeal to the IH.

($\leftarrow$): similarly. □

We remark that, using again terminating programs, the above Lemma covers immediate denial as well.

We now extend the effect of the above compilation on a program:

Definition 1.43

$$\begin{array}{rcll}
(\top)^{\vee_D} & = & \top & \\
(\bot)^{\vee_D} & = & \bot & \\
(\forall(Q \leftarrow G))^{\vee_D} & = & \forall(Q \leftarrow (G)^{\vee_G}) & \\
(D_1 \wedge D_2)^{\vee_D} & = & (D_1)^{\vee_D} \wedge (D_2)^{\vee_D} & \\
(D_1 \vee D_2)^{\vee_D} & = & D & \textit{where } (D_1)^{\vee_D} \vee (D_2)^{\vee_D} \setminus D \\
(X)^{\vee_G} & = & X & X \in \{\top, \bot, Q, \textit{(dis)eq}\} \\
(G_1 \wedge G_2)^{\vee_G} & = & (G_1)^{\vee_G} \wedge (G_2)^{\vee_G} & \\
(G_1 \vee G_2)^{\vee_G} & = & (G_1)^{\vee_G} \vee (G_2)^{\vee_G} & \\
(D \rightarrow G)^{\vee_G} & = & (D)^{\vee_D} \rightarrow (G)^{\vee_G} & \\
(\forall x : a.\, G)^{\vee_G} & = & \forall x : a.\, (G)^{\vee_G} &
\end{array}$$

Theorem 1.44 (Elimination of $\vee$) *For every ground substitution $\Gamma \vdash \theta : \Phi$,*

1. $\mathcal{S} :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta]G$ *iff* $\vee\mathcal{S} :: \Gamma; ([\theta]\mathcal{D})^{\vee_D} \vdash_{\mathcal{P}} ([\theta]G)^{\vee_G}$*;*
2. $\mathcal{I} :: \Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta]D \gg Q$ *iff* $\vee\mathcal{I} :: \Gamma; ([\theta]\mathcal{D})^{\vee_D} \vdash_{\mathcal{P}} ([\theta]D)^{\vee_D} \gg Q$*.*

Proof: ($\rightarrow$). By a straightforward mutual induction on the structure of the given derivations: we show the crucial case.

Case: $\mathcal{I}$ ends in $\gg\vee$:

$\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta]D_1 \gg Q$ and $\Gamma; \mathcal{D} \vdash_{\mathcal{P}} [\theta]D_2 \gg Q$ By sub-derivation

$\Gamma; ([\theta]\mathcal{D})^{\vee_D} \vdash_{\mathcal{P}} ([\theta]D_1)^{\vee_D} \gg Q$ and $\Gamma; ([\theta]\mathcal{D})^{\vee_D} \vdash_{\mathcal{P}} ([\theta]D_2)^{\vee_D} \gg Q$ By IH 2

$\Gamma; ([\theta]\mathcal{D})^{\vee_D} \vdash_{\mathcal{P}} ([\theta]D_1)^{\vee_D} \vee ([\theta]D_2)^{\vee_D} \gg Q$ By rule $\gg\vee$

$\Gamma; ([\theta]\mathcal{D})^{\vee_D} \vdash_{\mathcal{P}} [\theta](D_1 \vee D_2)^{\vee_D} \gg Q$ By Lemma ??

($\leftarrow$). Similarly. □

The same remark w.r.t. termination applies.

We finish up this Chapter by completing our two running example, where in a final pass we have also:

- Renamed negative predicates with more suggestive names.
- Removed irrelevant occurrences of $\top$.
- Hidden the irrelevant augmentation in the final negative clause.
- Brought clauses to the 'core' languages syntax, as in LF.

Example 1.45 *Concluding Example 6.26 and ?? the final definition of* $\neg$**`closed`** *is:*

$$\begin{array}{rcl}
\textit{oplam} & : & \forall E{:}\textit{exp}.\, \textit{open}\ (\textit{lam}\ E) \leftarrow \forall x{:}\textit{exp}.\ \textit{open}\ (E\ x). \\
\textit{opap1} & : & \forall E_1, E_2{:}\textit{exp}.\, \textit{open}\ (\textit{app}\ E_1\ E_2) \leftarrow \textit{open}\ E_1. \\
\textit{opap2} & : & \forall E_1, E_2{:}\textit{exp}.\, \textit{open}\ (\textit{app}\ E_1\ E_2) \leftarrow \textit{open}\ E_2.
\end{array}$$

where we have hidden the irrelevant assumption closed x in `oplam` *and renamed '*$\neg$`closed`*' into* `open`*.*

Example 1.46 *The final definition of* $\neg$ **linear** *and in turn* $\neg$ **linx** *is:*

$$
\begin{array}{rcl}
\neg linapp1 & : & \neg linear(app\ E_1\ E_2) \leftarrow \neg linear(E_1). \\
\neg linapp2 & : & \neg linear(app\ E_1\ E_2) \leftarrow \neg linear(E_2). \\
\neg linlam1 & : & \neg linear(lam\ \lambda x\,.\,E\ x) \leftarrow \neg linx(\lambda x\,.\,E\ x) \\
.\neg linlam2 & : & \neg linear(lam\ \lambda x\,.\,E\ x) \leftarrow (\forall y{:}exp.\,(\neg linx(\lambda x\,.\,y) \wedge linear(y)) \rightarrow \neg linear(E\ y)). \\
\neg linxap0 & : & \neg linx(\lambda x\,.\,app\ (E_1\ x^1)\ (E_2\ x^1)). \\
\neg linxap1 & : & \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x^1)) \leftarrow \neg linx(\lambda x\,.\,E_1\ x). \\
\neg linxap2 & : & \neg linx(\lambda x\,.\,app\ (E_1\ x^1)\ (E_2\ x)) \leftarrow \neg linx(\lambda x\,.\,E_2\ x). \\
\neg linxap3 & : & \neg linx(\lambda x\,.\,app\ (E_1\ x)\ (E_2\ x)) \leftarrow \neg linx(\lambda x\,.\,E_1) \wedge \neg linx(\lambda x\,.\,E_2\ x). \\
\neg linxlm & : & \neg linx(\lambda x\,.\,lam(\lambda y\,.\,E\ x\ y)) \leftarrow \\
 & & \qquad (\forall y{:}exp.\,\neg linx(\lambda x\,.\,y) \rightarrow \neg linx(\lambda x\,.\,E\ x\ y)).
\end{array}
$$

1.10 Summary

In this Chapter we have given a complement algorithm for a significant fragment of third order Hereditary Harrop Formulae, by adapting the idea of *elimination* of negation introduced in [?] for Horn logic. This has the neat effect that negation and its problems are *eliminated*, i.e. we avoid any extension to the (meta) language. This has entailed finding a middle ground between the Closed World Assumption usually associated with negation and the Open World Assumption typical of logical frameworks. Our solution is to restrict the set of programs we deem deniable in a novel way, so as to enforce a *Regular Word Assumption* (RWA): we define a class of programs whose dynamic assumptions extend the current database in a specific regular way. Technically, this regularity under dynamic extension is calibrated so as to ensure that static and dynamic clauses never *overlap*. This property extends to the negative program; in a sense, we maintain a distinction between static and dynamic information, but at a much finer level, i.e. *inside* the definition of a predicate. The resulting fragment is very rich, as it captures the essence of the usage of hypothetical and parametric judgments in a logical framework; namely, that they are intrinsically combined to represent *scoping* constructs in the object language.

Chapter 2

Conclusions and Future Work

The importance of higher-order logical frameworks and logic programming languages that depend on intuitionistic logic, open-world assumptions (changing contexts), and lambda-abstractions is becoming more apparent and their use more widespread. Recent research is attempting to increase their expressivity, while preserving the conciseness and elegance of their representation techniques. The issue of negation has to be appreciated in that context. A good understanding of negation in such settings will significantly enhance the expressive strengths of such specification languages.

In this dissertation we have presented a solution to this long-standing issue in logical frameworks endowed with a logic programming interpretation. The solution offered by our approach has the net (and neat) effect that negation and its problems are *eliminated*, i.e. we avoid any extension to the (meta) language.

Although the transformational approach to negation has been investigated in traditional logic programming, this is a novel approach to addressing negative information in the higher-order intuitionistic setting. Many of the techniques from Horn programs do not carry over directly, so creative solutions had to be found to adapt the idea to the higher-order setting. In particular, we had to:

1. Formulate a strict λ-calculus which is closed under term complement.
2. Find a notion of negation normal forms which is compatible with the operational semantics required by Hereditary Harrop Formulae (HHF).
3. Introduce the *Regular World Assumption* (RWA) as a way to reconcile the intrinsic tension between the *Closed World Assumption*, associated with negation, and the *Open World Assumption* typical of languages with embedded implication.

Elimination of negation is particularly tuned to logical frameworks: although the problems connected with negation are analogous to logical frameworks and logic programming, the solution does not need to be the same. In this sense out approach is not a panacea. For example, it is definitely non appropriate in presence of even a small database of facts. Although the typing discipline goes a long way to limit the combinatorial explosion of negative facts, often a different approach is more fruitful; consider a small database recording the age of some people. It would be painful to negate say `age k` $\mathtt{s}^{34}$`(0)` by inferring `non-age k 0`...`non-age k` $\mathtt{s}^{33}$`(0), non-age k` $\mathtt{s}^{35}$`(X)` . In this case the use of some form of *constructive* negation is preferable, possibly in the form of disequations, i.e. `non-age k Z` $\leftarrow$ `Z` $\neq$ $\mathtt{s}^{34}$`(0)`. This area, to date, has not been explored at all in the higher-order case.

Before discussing how to extend our approach beyond some of the current limitations, we take on again the main technical restriction, i.e. to *complementable* programs as defined in Figure 6.7[1]. As we have argued, the key to a successful and implementable pairing of negation and hypothetical judgments is to keep separate at any time static and dynamic information in a program. We have achieved this by requiring every assumption to be parametric, a property which is preserved by the negative program. The eigenvariable condition enforced by the operational semantics of HHF together with the rigidity restriction does the rest. We have argued that this restriction is a most natural one w.r.t. the intended application; in our experience, it covers the overwhelming majority of actual Twelf and λProlog code. This of course does not mean that

[1] We discuss the issue of local variables later, see Subsection ??.

there are no useful programs which lie outside of this class; it is indeed entirely possible to use hypothetical judgments in isolation from parametric ones. We have argued in Section 5.4 how sometimes this can be avoided by resorting to a finer notion of context such as the one offered in linear logic. Otherwise, it is always possible to eliminate the offending implications by introducing an explicit management of contexts. For instance, we can revisit Example 5.1 and transform

$$impi \quad : \quad nd(A\ imp\ B) \leftarrow (nd(A) \rightarrow nd(B)).$$

into:

$$\begin{aligned}
impi' &\ : \ nd(A\ imp\ B) \leftarrow nd2(A, B)\\
impi1 &\ : \ nd2(A, A).\\
impi2 &\ : \ nd2(C, A\ imp\ B) \leftarrow nd2(C, B)\\
impi3 &\ : \ nd2(C, A\ imp\ B) \leftarrow nd2(A, B).
\end{aligned}$$

Although this goes against the spirit of logical frameworks based on intuitionistic logic programming , which owes part of its success to the capability of representing object-logics contexts via the meta-level scoping mechanism of embedded implication, this is not unheard of, and it is actually proposed in [?]. Besides, this transformation will of course be localized only to those predicates we need to complement.

2.1 Lifting Restrictions

We now address some of the restrictions which can in fact be lifted; in doing so we will consider an example which is not currently treated:

Example 2.1 *Consider the following extension of the* `copy` *clauses to a third and fourth order constructs, respectively* `callcc` *and* `reset`*:*

$$\begin{aligned}
callcc &\ : \ ((exp \rightarrow exp) \rightarrow exp) \rightarrow exp.\\
reset &\ : \ (((exp \rightarrow exp) \rightarrow exp) \rightarrow exp) \rightarrow exp\\
cpcallcc &\ : \ cp\ (callcc\ \lambda c{:}exp \rightarrow exp.\, E\ c)\ (callcc\ \lambda d{:}exp \rightarrow exp.\, F\ d)\\
&\qquad \leftarrow (\forall c{:}exp \rightarrow exp.\\
&\qquad\qquad (\forall x{:}exp.\, \forall y{:}exp.\, cp\ (c\ x)\ (c\ y) \leftarrow cp\ x\ y)\\
&\qquad\qquad\qquad \rightarrow cp\ (E\ c)\ (F\ c)).\\
cpreset &\ : \ cp\ (reset\ \lambda f{:}(exp \rightarrow exp) \rightarrow exp.\, E\ f)\ (reset\ \lambda g{:}(exp \rightarrow exp) \rightarrow exp.\, F\ g)\\
&\qquad \leftarrow (\forall f{:}(exp \rightarrow exp) \rightarrow exp.\\
&\qquad\qquad (\forall c, d{:}exp \rightarrow exp.\\
&\qquad\qquad\qquad cp\ (f\ c)\ (f\ d))\\
&\qquad\qquad\qquad\qquad \leftarrow (\forall x, y{:}exp.\, cp\ x\ y \rightarrow cp\ (c\ x)\ (d\ y))\\
&\qquad\qquad\qquad\qquad\qquad \rightarrow cp\ (E\ f)\ (F\ f))
\end{aligned}$$

2.1.1 Parameters Restrictions

The restriction to parameters of base type seems to be the less complicated to lift, if we still require them to occur only in *head* position. We thus would redefine the class of Shallow Parameter Expressions (*SPE*) as:

$$SPE \quad e_x \quad ::= \quad x : A \mid \lambda x \,.\, e_x \mid (e_x\ M)^k$$

This will allow to complement the third-order *cpcallcc* clause:

$$c{:}exp \rightarrow exp \vdash \mathrm{Not}_{\alpha}(\forall x, y{:}exp.\, cp\ (c\ x)\ (c\ y) \leftarrow cp\ x\ y) =$$

$$\begin{array}{l}\forall(\neg cp\ (c\ x)\ (lam\ E) \wedge \neg cp\ (c\ x)\ (app\ E_1\ E_2) \wedge \\ \neg cp\ (lam\ F)\ (c\ y) \wedge \neg cp\ (app\ F_1\ F_2)\ (c\ y) \wedge \\ \neg cp\ (c\ x)\ (c\ y) \leftarrow \neg cp\ x\ y).\end{array}$$

The condition for parameter to occur at head position instead is orthogonal, since it is a sufficient condition for maintaining non-overlapping between dynamic and static clauses; that is, of course, the main technical idea behind the exclusivity proof. A clause can indeed be *complementable*, i.e. every assumption is parametric, but if the eigenvariable does not occur in head position in the assumption, then the non-overlapping requirement may not be immediately verifiable. If we can detect by static program analysis or by any other means that no overlapping will result, then those clauses too can be promoted.

2.1.2 Extension to Any Order

We currently treat only the third-order case, that is we allow HHF which only make Horn assumptions. In this way, generally speaking, judgments on goals were only trivially recursive with the ones on clauses, since the latter would not make any new assumption. Allowing arbitrary assumptions requires instead this recursion to be unbounded. In general, most of the times we simply need to modify those judgments by passing around their contexts. To be concrete, let us consider for instance the issue of *schema extraction*; in the n-ary case, schemata ought to be *hereditarely closed.* In fact, if a HHF belongs to a schema, the latter ought to take into account any further assumption that the former may yield. The aforementioned modifications of the extraction judgments in Figure 6.5 will do the trick:

$$\frac{\Gamma; \mathcal{D} \vdash G \overset{G}{\Longrightarrow} \mathcal{S}_1 \qquad \Gamma; \mathcal{D} \vdash D \overset{D}{\Longrightarrow} \mathcal{S}_2}{\Gamma; \mathcal{D} \vdash (G \to D) \overset{D}{\Longrightarrow} \mathcal{S}_1 \| \mathcal{S}_2} \overset{D}{\Longrightarrow}\to$$

$$\ldots$$

$$\frac{\Gamma; (\mathcal{D} \wedge D) \vdash G \overset{G}{\Longrightarrow} \mathcal{S}_1 \qquad \Gamma; \mathcal{D} \vdash D \overset{D}{\Longrightarrow} \mathcal{S}_2}{\Gamma; \mathcal{D} \vdash D \to G \overset{G}{\Longrightarrow} \mathcal{S}_1 \| \mathcal{S}_2} \overset{G}{\Longrightarrow}\to$$

For example we can extract the following schema from the extended definition of `cp`:

$$\begin{array}{lll}\mathrm{def}(cp) & \overset{D}{\Longrightarrow} & x : exp; cp\ x\ x \| \\ & & c : exp \to exp; (\forall x, y{:}exp.\, cp\ x\ y \to cp\ (c\ x)\ (c\ y)) \| \\ & & f : (exp \to exp) \to exp; (\forall c, d{:}exp \to exp.\, cp\ (f\ c)\ (f\ d) \leftarrow (\forall x, y{:}exp.\, cp\ x\ y \to cp\ (c\ x)\ (c\ y)))\end{array}$$

In this case only the schema alternative induced by the `cpreset` clause needs to be hereditarely closed with `x : exp; cp x x`; but this is already provided by the (schema extracted from the) `cplam` clause.

Similar changes apply to the other judgments; in particular, augmentation is generalized, so that goals which can make dynamic assumptions will be recursively augmented as well. Formally:

$$\frac{\Gamma; \mathcal{D} \vdash aug_{\mathrm{D}}(D) = D^a \qquad \Gamma; (\mathcal{D} \wedge D) \vdash aug_{\mathrm{G}}(G) = G^a}{\Gamma; \mathcal{D} \vdash aug_{\mathrm{G}}(D \to G) = D^a \to G^a}\, aug_{\mathrm{G}} \to$$

We can therefore complement a fourth-order clause such as `cpreset`, where, for the sake of readability, we have not expanded the calls to $\mathrm{Not}_{\alpha}(D)$:

$$\begin{array}{lll}\neg cpreset & : & \neg cp\ (reset\ \lambda f{:}(exp \to exp) \to exp.\, E\ f)\ (reset\ \lambda g{:}(exp \to exp) \to exp.\, F\ g) \\ & & \leftarrow (\forall f{:}(exp \to exp) \to exp. \\ & & \quad (\forall c, d{:}exp \to exp. \\ & & \qquad (\forall x, y{:}exp.\, cp\ x\ y \wedge \mathrm{Not}_{\alpha}(cp\ x\ y) \to \neg cp\ (c\ x)\ (d\ y)) \\ & & \qquad\quad \to \neg cp\ (f\ c)\ (f\ d) \wedge \mathrm{Not}_{\alpha}(cp\ (f\ c)\ (f\ d) \\ & & \qquad\qquad \to \neg cp\ (E\ f)\ (Ff))\end{array}$$

2.1.3 Open Queries

One of the more widespread criticism to negation-as-failure (*NF*) is that it cannot return answer substitutions; in fact, if it is not restricted to ground goals it may return unsound solutions, as detailed in Section 1.4. On the other hand, one of the bright spots of elimination of negation is the *positivization* of negative information, which can now be queried as regular (HHF) clauses. In particular, negative open queries are simply positive queries to the complement program and thus will return a witness if one exists. However, open queries are not accounted for in the theory we have developed so far. Indeed, we have used programs just as *decision procedures*, i.e. such that for every ground goal only yes/no answers can be returned. This does not need to be the case; open queries can be allowed in so far as enough 'directionality' is retained in the query. In fact, if we can split the arguments of a query in an *input* and *output* sequences so that termination (in the input elements) can still be determined, the proof of exclusivity will go through, provided that the no-overlapping requirement holds. However, this is almost infallibly the case, because parameters tend to occur in input position; in fact, as we have already remarked, parametric judgments are typically used to traverse scoping constructs and most of the times termination can be proven w.r.t. those terms. The idea is to give *modes* to predicate definitions: let $q : A_1 \to \ldots \to A_n \to \mathbf{o}$ and $\mathrm{mode}(q) = m_1, \ldots, m_n$ be a *mode* for q, where each m_i is either $+$ (input) or $-$ (output) [?]. More complex moding systems have been proposed in the literature about traditional logic programming , but this one may be pragmatically adequate for our purposes.Now, given m_q consisting of a sequence of $m \geq 1$ +'s and $n \geq 0$ -'s, if we can show that $\mathrm{def}(q)$ is terminating w.r.t. m_q and, in case it is a non-Horn definition, if $\mathrm{def}(q)$ is parametric in at least one position i such that $1 \leq i \leq m$, then the query $\forall \overline{x_m}.\, \exists \overline{y_n}.\, q\ \overline{x_m}\ \overline{y_n}$ is allowed. This can be recursively extended as usual to any goal (and clause). To make the discussion more concrete, let us return to the `copy` example. The results in Chapter 6 guarantees the following, for an appropriate context schema $\Gamma; \mathcal{D}$:

- $\forall x\!:\!exp. \forall y\!:\!exp$ either $\Gamma; \mathcal{D} \vdash cp\ x\ y$ or $\Gamma; \mathcal{D} \vdash \neg cp\ x\ y$.
- $\forall x\!:\!exp. \forall y\!:\!exp$ it is not the case $\Gamma; \mathcal{D} \vdash cp\ x\ y$ and $\Gamma; \mathcal{D} \vdash \neg cp\ x\ y$.

But of course we may be interested in *finding* a y such that the above holds. By noting that `copy` can be shown to be terminating only with one term instantiated, we can consider say x as input and y as output. Additionally, we can check that the non-overlapping condition still holds in this case; thus we can establish:

- $\forall x\!:\!exp$ either $\exists y\!:\!exp$ such that $\Gamma; \mathcal{D} \vdash cp\ x\ y$ or $\Gamma; \mathcal{D} \vdash \neg cp\ x\ y$.
- $\forall x\!:\!exp$ it not the case that $\exists y\!:\!exp$ such that $\Gamma; \mathcal{D} \vdash cp\ x\ y$ and $\Gamma; \mathcal{D} \vdash \neg cp\ x\ y$.

2.1.4 Local Variables Revisited

Allowing non-ground queries has sometimes the side-effect of overcoming the strict restriction to programs without local variables. In certain cases, in fact, we can lift the local variable to the head of the clause (and possibly embed it in a constructor); thus, the variable is no more (intrinsically) existentially quantified in the body and will not cause any problem during complementation; moreover, we do not change the arity of the predicate definition, although the latter too is possible. If we can still prove the program terminating, that is the local variable plays no role in the termination argument, then we can use that predicate definition leaving the ex-local variable as an output value only. For example, we can revisit the program for type-checking in the simply typed calculus; if we change application to allow the type of the argument to the function, that is:

$$
\begin{array}{rcl}
app & : & exp \to exp \to tp \to exp. \\
ofapp & : & of\ (app\ E_1\ E_2\ T_1)\ T_2 \\
 & & \quad \leftarrow of\ E_1\ (arrow\ T_1\ T_2) \\
 & & \quad \leftarrow of\ E_2\ T_1 \\
oflam & : & of\ (lam\ E)\ (arrow\ T_1\ T_2) \\
 & & \quad \leftarrow (\forall x\!:\!exp.\, of\ x\ T_1 \to of\ (E\ x)\ T_2).
\end{array}
$$

Then we can apply (a slight modification of) our complement algorithm, yielding:

$$
\neg app1 \quad : \quad \neg of\ (app\ E_1\ E_2\ T_1)\ T_2
$$

$$
\begin{array}{rcl}
 & & \leftarrow \neg of\ E_1\ (arrow\ T_1\ T_2). \\
\neg app2 & : & \neg of\ (app\ E_1\ E_2\ T_1)\ T_2 \\
 & & \leftarrow of\ E_1\ (arrow\ T_1\ T_2) \\
 & & \leftarrow \neg of\ E_2\ T_1. \\
\neg aug_{\mathrm{D}}(oflam) & : & \neg of\ (lam\ E)\ (arrow\ T_1\ T_2) \\
 & & \leftarrow (\forall x{:}exp.\ of\ x\ T_1 \rightarrow \\
 & & \quad (\forall T{:}tp.\ \neg of\ x\ T \leftarrow T \neq T_1) \rightarrow \\
 & & \qquad \neg of\ (E\ x)\ T_2).
\end{array}
$$

Of course, this lifting of local variables is not sound in general; in the former case, it is doable because T_1 remains in output position, as (possibly higher-order) mode analysis guarantees. On the other hand, for example, it will not work for predicates as graph reachability:

$$
\begin{array}{l}
path(X,Y) \leftarrow edge(X,Y). \\
path(X,Y) \leftarrow edge(X,Z) \wedge path(Z,Y).
\end{array}
$$

Here, the local variable Z becomes an input argument to the recursive call for `path`. It is our opinion that the issue of local variables during complementation does not have a universal solution. Approaches which embrace the extensional nature of universal quantification brought in by the negation of existential quantifiers [?, ?] are not satisfactory and robust enough to carry over to logical frameworks with intensional universal quantification. On the other end of the spectrum, the idea to simply keep those universally quantified predicates as constraints, such as in [?] will pay some run-time overhead. We instead conjecture that (sound) variable lifting may be enough for a sufficiently interesting class of programs w.r.t. our intended application. In fact, it goes along well with the rigidity restriction we have already to enforce to obtain exclusivity of complementation.

2.2 Extensions

The techniques we have described in this dissertation applies to a framework such as L_λ; we briefly comment on how to extend elimination of negation to richer languages.

2.2.1 Beyond Patterns

Term complementation and intersection require higher-order patterns; both λProlog and Twelf allow any (well-typed) terms, although they are dealt with differently. Since we have (dis)equations in our language, we can apply an idea analogous to normalization of input variables 6.4.1 and replace occurrence of non-patterns with fresh variables in those predicates we intend to complement, and constrain those variable to be equal to the replaced term. This is close to what happens in the Twelf implementation and allow the constraints handling mechanism to work its magic. If the framework is not constraint-oriented, it is well-known how to compile away the offending non-pattern occurrence via appropriate substitution predicates implemented through the `copy` clauses [?].

2.2.2 Richer Type Theories

The approach we have chosen is tailored to satisfy the requirements of more complex logical frameworks than L_λ; thus, we now mark some observations on how elimination of negation can be extended to richer type theories.

Polymorphism

Different degrees (in the λ-cube) of polymorphism have been advocated as a feature in logical frameworks. Of course, the more expressive the type theory, the more complicated its meta-theory. Pfenning has given an algorithm for patterns unification and generalization in the Calculus of Constructions [?]. Even if we stick to the fully applied case, term complementation may be fairly difficult to achieve in general. Languages such as λProlog instead offer the more manageable prenex (ML-like) polymorphism. In this case, a version of term complementation able to deal with polymorphic constructors, such as a polymorphic *cons*, should be feasible. In many ways, the problems are analogous to negation in presence of predicate quantification (see next entry ??), namely the tension between a static operation as complementation and the possible instantiations offered by polymorphism.

Dependent Types

Almost all the design decisions we have taken while addressing the issue of negation in a logical frameworkhave been motivated by the ease to extend the latter to a framework as LF/ *Twelf*. The very idea to allow negation by elimination owes to the attempt to preserving the adequacy of the extraordinary representation power of dependent types, while at the same time avoid to interfering with the underlying logic programming engine, which makes *Twelf* a unique unified meta-language for the theory and meta-theory of deductive systems. While we fall short in this dissertation to addressing dependent types directly, we believe that the machinery we have developed is robust enough for this extension. One novel problem that we can already foresee is related to the interaction between term complementation and *empty* types. In the simply-typed fragment we can assume every type to be inhabited, but this property is obviously undecidable in the more powerful setting. This turns out to be problematic when complementing variables; at first sight, it not clear what the complement of $\Gamma \vdash \mathrm{Not}(E\ \overline{x_n}) : a\ M$ would be. One possibility is to restrict variable complementation at type $a\ M$, perhaps such that M has no internal structure at all, i.e. it is empty or just a term variable; this would cover all the examples in this dissertation. Another one could be 'approximating' complementation in the simply-typed fragment and then sift out the resulting complement set, in view of dependencies.

2.2.3 Predicate Quantification

In the logic programming community the term 'higher-order ' is usually identified with the possibility of quantifying on predicates, in the effort to simulate the first-class functions capability of functional programming languages. This is a sometimes a source of misunderstandings (let me mention *Hilog* [?] which has a higher-order syntax and allows arbitrary terms to appear in places where predicates, functions and atomic formulas occur in predicate calculus, but its semantics is strictly first-order) and it may be overrated, at least as far as logical frameworks are concerned, as the success of frameworks such as those based on LF testify. Nevertheless, this is a useful, though not essential, feature and has been utilized for example in implementation of proof-carrying code with λProlog [?]. Although we have not investigated the issue in depth, it seems that clause complementation can be sometimes applied in this extended sense. For instance, let `some` be a predicate of type $(nat \to \mathbf{o}) \to natlist \to \mathbf{o}$, which selects the first element in a list of numbers for which a given predicate holds $P : nat \to \mathbf{o}$:

$$\begin{array}{rcl} shd & : & some\ (\lambda x{:}nat.\, P\ x)\ (cons\ y\ ys) \\ & & \quad \leftarrow (P\ y). \\ stl & : & some\ (\lambda x{:}nat.\, P\ x)\ (cons\ y\ ys) \\ & & \quad \leftarrow some\ (\lambda x{:}nat.\, P\ x)\ ys. \end{array}$$

The application of our algorithm would yield:

$$\begin{array}{rcl} \neg snil & : & \neg some\ (\lambda x{:}nat.\, P\ x)\ nil \\ \neg stl & : & \neg some\ (\lambda x{:}nat.\, P\ x)\ (cons\ y\ ys) \\ & & \quad \leftarrow \mathrm{Not}_{\mathrm{G}}(P\ y) \wedge \neg some\ (\lambda x{:}nat.\, P\ x)\ ys. \end{array}$$

Since we cannot foresee the structure of the goal $(P\ y)$ at compile-time, we delay the computation of $\mathrm{Not}_{\mathrm{G}}(P\ y)$ until the instantiation is known. Of course, we cannot allow unrestricted instantiations, but we need to restrict P to conform to the possible context schema. This will prevent goals such as $some\ (\lambda x : nat.\, \neg even(x) \to even(s(s(x)))\ (cons\ 0\ nil)$, which will destroy exclusivity.

Of course, the above example is too simple-minded in at least one respect; we did not define term complementation on terms with some internal logical structure. The only reason the `some` predicate was complementable is because the predicate occurring inside a term is simply a variable, making term complement trivial. In the general case, clause heads can contain arbitrary complex terms of type $\mathbf{o}$. The approach we have developed so far does not seem to capture those phenomena except in the simplest form.

2.3 Implementation Issues

A strict logical framework can be directly implemented with very minor adaptations of well-known techniques used for its linear cousins such as *Lolli* [?, ?] and *LLF* [?]. Although we argue that strictness is a useful and ubiquitous concept which deserves to be offered as a *primitive* notion in a logical framework, this is not the only choice.

In fact, Girard would be quick to point out that it not necessary to take strictness as a primitive at all, since linear logic is flexible enough to express the notion of 'must occur' already. Indeed, strict implication can be embedded into linear logic by simply defining $A \xrightarrow{1} B$ as $A \multimap A \to B$. This is of course true, but notice that while this translation will indeed retain provability, it not faithful to the structure of proofs. Since in a logical framework as LF we are also concerned with the structure of terms, this embedding is not adequate. For example, the strict term $\lambda x^1 .\, c\; x^1\; x^1$ corresponds to both $\lambda x^g \lambda y^u . c\; x^g\; y^u$ and $\lambda x^g \lambda y^u . c\; y^u\; x^g$, where the $(\_)^g$ notation refers to linear abstraction and application. Even if we are just considering proof search (and not terms) there are too many distinct derivations of $A \multimap A \to B$ when compared to $A \xrightarrow{1} B$. So when we take a theorem in strict logic, embed it, and run a logical frameworksuch as *LLF*, we incur into a fair amount of additional non-determinism. On the other hand, the strict λ-calculus captures exactly the right properties in an elegant way and can be developed from first principles. In summary, the linear λ-calculus here does not apply; even though it is clearly possible to compile strict functions to linear ones, this compilation preserves only truth, but not the structure of proofs.

Moreover, as far as negation is concerned, we can safely remain in an intuitionistic setting. The drawback is that we have to decorate source programs whose clause heads (hereditarely) contain partially applied terms with appropriate occurrences of the `vacuous` predicate we have mentioned in Section 2.5. The latter, under negation, is transformed in the `strict` one. Strict unification does not need to be considered, since every term remains in the fully-applied fragment.

For example, if we are encoding a term say $\ulcorner \Lambda x.\, e \urcorner$ with the side condition that $x \notin FV(e)$, we usually represent e with a pattern variable E, which does *not* depend on x. For example, reconsider clause `linxap1`:

$$linxap1 \quad : \quad linx(\lambda x \,.\, app\ (E_1\ x)\ E_2) \leftarrow linx(\lambda x \,.\, E_1\ x).$$

The latter may be rewritten as:

$$linxap1' \quad : \quad linx(\lambda x \,.\, app\ (E_1\ x)\ (E_2\ x)) \leftarrow vacuous(\lambda x \,.\, E_2\ x) \wedge linx(\lambda x \,.\, E_1\ x).$$

where $vacuos(\lambda x \,.\, E_2\ x)$ enforces that x does not occur in $E_2\ x$. The negation transformation will convert those annotations in (pre-compiled) `strict` ones. Pursuing further this example, the complement of `linxap1'` will include:

$$\neg linxap1' \quad : \quad \neg linx(\lambda x \,.\, app\ (E_1\ x)\ (E_2\ x)) \leftarrow strict(\lambda x \,.\, E_2\ x) \wedge \neg linx(\lambda x \,.\, E_1\ x).$$

This kind of decoration of programs is a relatively small price to pay compared to the hassle of implementing a strict calculus, only for the purpose of allowing full clause complementation; consider for example the lack so far of a crucial ingredient such as the type reconstruction algorithm (although one for a related calculus is presented in Wright's dissertation [?]). Moreover, for every signature, the definition of the predicate `vacuous` is completely trivial, while the one for `strict` is type-directed and can be automatically inferred, in the style of Miller's `copy` clause [?]. On the other hand, this approach is less workable when lifting the restriction to *ground* goals; in the presence of open queries, in fact, those annotations would induce an undesirable 'generate-and-test' operational semantics: for example, the `vacuous` predicate would generate vacuous terms to be then checked for, in this case, linearity. It may be possible to internalize the strictness annotations as *boolean constraints* [?], in analogy to the linear case; if so, that would work very well with logical frameworks such as *Twelf(X)* [?] which have a declarative notion of constraints.

2.4 Additional Topics

2.4.1 Higher-Order Program Algebra

Several authors have investigated the algebra of logic programs to address modularity and meta-level issues [?, ?, ?]. In particular in [?] the authors describe a program algebra for Horn clauses without local variables under clause complement, set union and intersection. Since they interpret negation as finite failure, exhaustivity does not hold. In fact, what they call a "constructive version of a boolean algebra" is exactly what Rasiowa [?] calls "quasi pseudo Boolean algebras", that is distributive lattices which satisfy the axioms of strong negation restricted to Horn logic.

We can extend the idea of logic program algebra to a significant fragment of (third-order) HHF. In our case we take equality ($\simeq$) as operational equivalence under appropriate run-time contexts [?]; for every program $\mathcal{P}_1, \mathcal{P}_2$ (seen as conjunction of predicate definitions) which satisfy a common context schema $\mathcal{S}$, for every run-time context $\Gamma; \mathcal{D} < \mathcal{S}$ and for every ground G:

$$\mathcal{P}_1 \simeq \mathcal{P}_2 \quad \text{iff} \quad (\Gamma; \mathcal{D} \vdash_{\mathcal{P}_1} G \text{ iff } \Gamma; \mathcal{D} \vdash_{\mathcal{P}_2} G)$$

Then we can organize the set of (finite) programs into a boolean algebra under union '$\wedge$' (lub), intersection '$\vee$' (glb), complement '$\mathrm{Not_D}$', empty program '$\top$' (zero) and universal program '$\bot$' (one). Corollary **??** confirms that negation is boolean. Moreover, the rules for $\mathrm{Not_G}, \mathrm{Not_D}$ have been engineered to respect De Morgan's laws.

It could be interesting to study the applicability of those ideas to modularity of higher-order logic programs, in particular to the modular construction of knowledge-based systems; for example a user could collect in a module $\mathcal{P}_1$ all the positive knowledge about a predicate and in another, say $\mathcal{P}_2$, all the negative ones (that is, the cases where the predicate does not hold). This is useful when dealing with defaults and exceptions. The system would be able to compose those modules via boolean manipulations.

2.4.2 Strictness in Explicit Substitutions

It is our contention that the strict λ-calculus that we have introduced to formulate term complementation has an independent interest in the investigation of sub-structural logics. Not only our types system is simpler and (we claim) more elegant of the ones presented in the literature (reviewed in Section 3.3), but the introduction of the notion of *vacuous* variables can be useful in a variety of contexts, beyond strictness analysis. One example is the study of *explicit substitutions* [?] in resource-conscious λ-calculi: for example, in [?] and refined in [?], the authors propose a system of explicit substitutions for intuitionistic linear logic over unit, lollipop, tensor and bang, where variables can either be linear or intuitionistic. The calculus is not optimal for several reasons, to start with the need for commutative conversions. One technical issue which could be improved is the substitution "extension operator" which accounts for term to be substituted and comes in three flavors: intuitionistic, linear and 'used' linear. to mark a term which has already been linearly substituted. We conjecture that the availability of a well-behaved theory of vacuous variables can offer a simpler analysis of the above point.

Bibliography

[AAB92] J. M. Dunn A. Anderson and N.D. Belnap. *Entailment. The Logic of Relevance and Necessity*, volume 2. Princeton University Press, 1992.

[AAB+95] Johan Andersson, Stefan Andersson, Kent Boortz, Mats Carlsson, Hans Nilsson, Thomas Sjöland, and Johan Widen. SICStus prolog user's manual. Technical Report T93-01, Swedish Institute of Computer Science, 1995.

[AB75] A. Anderson and N.D. Belnap. *Entailment. The Logic of Relevance and Necessity*, volume 1. Princeton University Press, 1975.

[AB90] Krzysztof R. Apt and Marc Bezem. Acyclic programs. In Peter Warren, David H.D.; Szerdei, editor, *Proceedings of the 7th International Conference on Logic Programming (ICLP '90)*, pages 617–633, Jerusalem, June 1990. MIT Press.

[AB94] K. Apt and R. Bol. Logic programming and negation. *Journal of Logic Programming*, 19/20:9–72, May/July 1994.

[ABT90] D. Pedreschi A. Brogi, P. Mancarella and F. Turini. Universal quantification by case analysis. In *Proc. ECAI-90*, pages 111–116, 1990.

[ABW88] K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases*, chapter 2, pages 89–148. Morgan Kaufmann, 1988.

[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[Acz77] Peter Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.7, pages 739–782. North-Holland, Amsterdam, 1977.

[AF99] Andrew W. Appel and Amy P. Felty. Lightweight lemmas in lambda prolog. In Danny De Schreye, editor, *Proceedings of the International Conference on Logic Programming (ICLP'99)*, pages 411–425, Las Cruces, New Mexico, December 1999. MIT Press.

[Apt90] Krzysztof R. Apt. Logic programming. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 10, pages 493–574. The MIT Press, New York, N.Y., 1990.

[Bar80] H. P. Barendregt. *The Lambda-Calculus: Its Syntax and Semantics*. North-Holland, 1980.

[BDLM00] Bugliesi, Delzanno, Liquori, and Martelli. Object calculi in linear logic. *JLC: Journal of Logic and Computation*, 10, 2000.

[Bel74] N.D. Belnap. Functions which really depend on their argument. Manuscript, 1974.

[Bel93] N. D. Belnap. Life in the undistributed middle. In K. Dosen and P. Schroeder-Heister, editors, *Substructural Logics*, pages 31–42. Oxford University Press, 1993.

[BF93] Clement A. Baker-Finch. Relevance and contraction: A logical basis for strictness and sharing analysis. Technical Report ISE RR 34/94, University of Canberra, 1993.

[BLLM94] P. Bruscoli, F. Levi, G. Levi, and M. C. Meo. Compilative constructive negation in constraint logic programs. In Sophie Tiso, editor, *Proc. Trees in Algebra and Programming - CAAP'94, 19th International Colloquium*, volume 787, pages 52–76. Springer, 1994.

[BM90] Anthony J. Bonner and L. Thorne McCarty. Adding negation-as-failure to intuitionistic logic programming. In Saumya Hermenegildo Manuel Debray, editor, *Proceedings of the 1990 North*

American Conference on Logic Programming, pages 671–693, Austin, TX, October 1990. MIT Press.

[BMPT87] R. Barbuti, Paolo Mancarella, Dino Pedreschi, and Franco Turini. Intensional negation of logic programs. In *Proceedings ofTapsoft87*, volume 259 of *LNCS*, pages 96–110, Austin, TX, October 1987. Springer Verlag Press.

[BMPT90] Roberto Barbuti, Paolo Mancarella, Dino Pedreschi, and Franco Turini. A transformational approach to negation in logic programming. *Journal of Logic Programming*, 8:201–228, 1990.

[BMV89] A. J. Bonner, L. T. McCarty, and K. Vadaparty. Expressing Database Queries with Intuitionistic Logic. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings of the North American Conference on Logic Programming*, pages 831–850, Cleveland, Ohio, USA, 1989.

[Bol88] A. W. Bollen. *Conditional Logic Programming.* PhD thesis, Australian National University, 1988.

[Bol90] A. W. Bollen. Relevant logic programming. *Journal of Automated Reasoning*, 7(4):563–586, December 1990.

[Bon91] A.J. Bonner. *Hypothetical Reasoning in Deductive Databases.* PhD thesis, Rutgers University, October 1991.

[Bon94] A.J. Bonner. Hypothetical reasoning with intuitionistic logic. In R. Demolombe and T. Imielinski, editors, *Non-Standard Queries and Answers*, volume 306 of *Studies in Logic and Computation*, pages 187–219. Oxford University Press, 1994.

[BR57] A. Bialynicki-Birula and H. Rasiowa. On the representation of quasi-boolean algebras. *Bulletin de L'académie Polonaise des Sciences*, 5:259–261, 1957.

[CdPR99] Iliano Cervesato, Valeria de Paiva, and Eike Ritter. Explicit substitutions for linear logical frameworks: Preliminary results. In A. Felty, editor, *Proceedings of the Worskshop on Logical Frameworks and Meta-Languages — LFM'99*, Paris, France, 28 September 1999.

[Cer96] Iliano Cervesato. *A Linear Logical Framework.* PhD thesis, Dipartimento di Informatica, Università di Torino, February 1996.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic.* North-Holland, Amsterdam, 1958.

[Cha88] D. Chan. Constructive negation based on the completed databases. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth International Conference and Symposium on Logic Programming*, pages 111–125, Seattle, Washington, August 15-19 1988.

[Cha89] D. Chan. An Extension of Constructive Negation and its Application in Coroutining. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings of the North American Conference on Logic Programming*, pages 477–496, Cleveland, Ohio, USA, 1989.

[CHP97] Iliano Cervesato, Joshua S. Hodas, and Frank Pfenning. Efficient resource management for linear logic proof search. *Theoretical Computer Science*, 1997. To appear in a special issue on Proof Search in Type-Theoretic Languages, D. Galmiche, editor.

[Chu41] A. Church. *The Calculi of Lambda Conversion.* Princeton University Press, 1941.

[Chu51] A. Church. The weak theory of implication. In *Kontrolliertes.* Karl Albert, 1951.

[CKW93] Weidong Chen, Michael Kifer, and David S. Warren. Hilog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, February 1993.

[CL89] Hubert Comon and Pierre Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7(3–4):371–425 (or 371–426??), March–April 1989.

[Cla78] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, New York, 1978.

[Col84] A. Colmerauer. Equations and Inequations on Finite and Infinite Trees. In *Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS-84)*, pages 85–99, Tokyo, Japan, November 1984. ICOT.

[Com88] H. Comon. *Unification et disunification. Théories et applications.* Thèse de Doctorat d'Université, Institut Polytechnique de Grenoble (France), 1988.

[Com91] H. Comon. Disunification: a survey. In J-L. Lassez and G.Plotkin, editors, *Computational Logic.* MIT Press, Cambridge,MA, 1991.

[Com98] H. Comon. About proofs by consistency. *Lecture Notes in Computer Science*, 1379:136–??, 1998.

[CP96] Iliano Cervesato and Frank Pfenning. A linear logical framework. In E. Clarke, editor, *Proceedings of the Eleventh Annual Symposium on Logic in Computer Science*, pages 264–275, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.

[dB80] N.G. de Bruijn. A survey of the project AUTOMATH. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, 1980.

[dB91] N.G. de Bruijn. A plea for weaker frameworks. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 40–67. Cambridge University Press, 1991.

[DM97] Raymond Dowell and Dale Miller. A logic for reasoning with higher-order abstract syntax: An extended abstract. In Glynn Winskel, editor, *Proceedings of the Twelfth Annual Symposium on Logic in Computer Science*, pages 434–445, Warsaw, Poland, June 1997.

[DPS97] Joëlle Despeyroux, Frank Pfenning, and Carsten Schürmann. Primitive recursion for higher-order abstract syntax. In R. Hindley, editor, *Proceedings of the Third International Conference on Typed Lambda Calculus and Applications (TLCA'97)*, pages 147–163, Nancy, France, April 1997. Springer-Verlag LNCS. An extended version is available as Technical Report CMU-CS-96-172, Carnegie Mellon University.

[Dun92] P. M. Dung. Declarative semantics of hypothetical logic programming with negation as failure. In E. Lamma and P. Mello, editors, *Proceedings of the Third International Workshop on Extensions of Logic Programming (ELP '92)*, volume 660 of *LNAI*, pages 45–58, Bologna, Italy, February 1992. Springer Verlag.

[Eri93] Lars-Henrik Eriksson. *Finitary Partial Inductive Definitions and General Logic*. PhD thesis, Department of Computer and System Sciences, Royal Institute of Technology, Stockholm, 1993.

[Fag97] François Fages. Constructive negation by pruning. *Journal of Logic Programming*, 32(2):85–118, August 1997.

[FBM93] G. Levi F. Bruscoli, F. Levi and M.C. Meo. Intensional negation in constraint logic programming. In D. Sacca, editor, *Proc. GULP93*, pages 359–373, 1993.

[FRTW88] Norman Foo, Anand Rao, Andrew Taylor, and Adrian Walker. Deduced relevant types and constructive negation. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 126–139, Seatle, 1988. ALP, IEEE, The MIT Press.

[Gab85] Dov M. Gabbay. N-Prolog: An extension of Prolog with hypothetical implications II. Logical foundations and negation as failure. *Journal of Logic Programming*, 2(4):251–283, December 1985.

[Gab91] Dov Gabbay. Modal provability foundations for negation by failure, in extensions of logic programming. In Peter Schroeder-Heister, editor, *Extensions of Logic Programming: International Workshop, Tübingen FRG, December 1989*, volume 475 of *Lecture Notes in Artificial Intelligence*, pages 179–222. Springer-Verlag, 1991.

[GdQ92] Dov M. Gabbay and Ruy J. G. B. de Queiroz. Extending the Curry-Howard interpretation to linear, relevant and other resource logics. *The Journal of Symbolic Logic*, 57(4):1319–1365, December 1992.

[GH78] J. Guttag and J. Horning. The algebraic specification of abstract data types. *Acta Informatica*, 10(1):27–52, 1978.

[GL90] Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In David H.D. Szerdei Peter Warren, editor, *Proceedings of the 7th International Conference on Logic Programming (ICLP '90)*, pages 579–597, Jerusalem, June 1990. MIT Press.

[GMR92] L. Giordano, A. Martelli, and G. Rossi. Extending Horn clause logic with implication goals. *Theoretical Computer Science*, 95(1):43–74, March 1992.

[GO98] Laura Giordano and Nicola Olivetti. Negation as failure and embedded implication. *Journal of Logic Programming*, 36(2):91–147, August 1998.

[GPR98] Neil Ghani, Valeria De Paiva, and Eike Ritter. Linear explicit subsitutions. Technical Report CSR-98-2, University of Birmingham, School of Computer Science, March 1998.

[GR84] Dov M. Gabbay and U. Reyle. N-Prolog: An extension of Prolog with hypothetical implications I. *Journal of Logic Programming*, 1(4):319–355, December 1984.

[GR87] Jean H. Gallier and Stan Raatz. Hornlog: A graph-based interpreter for general Horn clauses. *Journal of Logic Programming*, 4(2):119–155, June 1987.

[GS86] Dov M. Gabbay and Marek J. Sergot. Negation as inconsistency I. *Journal of Logic Programming*, 3(1):1–35, April 1986.

[Hal91] L. Hallnas. Partial inductive definitions. *Theoretical Computer Science*, 87(1):115–147, July 1991.

[Har89] J. Harland. A Kripke-like Model for Negation as Failure. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings of the North American Conference on Logic Programming*, pages 626–644, Cleveland, Ohio, USA, 1989.

[Har91a] James Harland. A clausal form for the completion of logic programs. In Koichi Furukawa, editor, *Proceedings of the 8th International Conference on Logic Programming*, pages 711–725. MIT, June 1991.

[Har91b] James Harland. *On Hereditary Harrop Formulae as a Basis for Logic Programming*. PhD thesis, Edinburgh, January 1991.

[Har92] James Harland. On normal forms and equivalence for logic programs. In Krzysztof Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 146–160, Washington, USA, 1992. The MIT Press.

[Har93] James Harland. Success and failure for hereditary Harrop formulae. *Journal of Logic Programming*, 17(1):1–29, October 1993.

[Hel77] G. Helmann. Completeness of the normal typed fragment of the λ-system u. *Journal of Philosophical Logic*, 1977.

[Hey56] A. Heyting. *Intuitionism, an Introduction*. North-Holland, Amsterdam, 3 edition, 1956.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

[HM90] Joshua S. Hodas and Dale Miller. Representing objects in a logic programming language with scoping constructs. In Peter Warren, David H.D.; Szerdei, editor, *Proceedings of the 7th International Conference on Logic Programming (ICLP '90)*, pages 511–528, Jerusalem, June 1990. MIT Press.

[HM94] Joshua Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994. A preliminary version appeared in the Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science, pages 32–42, Amsterdam, The Netherlands, July 1991.

[Hod94] Joshua S. Hodas. *Logic Programming in Intuitionistic Linear Logic: Theory, Design, and Implementation*. PhD thesis, University of Pennsylvania, Department of Computer and Information Science, 1994.

[HP96] M. Hanus and C. Prehofer. Higher-order narrowing with definitional trees. In *Proc. Seventh International Conference on Rewriting Techniques and Applications (RTA'96)*, pages 138–152. Springer LNCS 1103, 1996.

[HP97] J. Harland and D. Pym. Resource-distribution via Boolean constraints. In William McCune, editor, *Proceedings of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 222–236, Berlin, July 13–17 1997. Springer.

[HSH90] L. Hallnas and P. Schroeder-Heister. A proof-theoretic approach to logic programming: Clauses as rules. *Journal of Logic and Computation*, 1(2):635-660, October 1990.

[HSH91] L. Hallnas and P. Schroeder-Heister. A proof-theoretic approach to logic programming: Programs as definitions. *Journal of Logic and Computation*, 1(5):261-283, October 1991.

[Isa98] Isabelle. System home page, October 1998. Version 98-1.

[Jen91] Thomas P. Jensen. Strictness Analysis in Logical Form. In John Hughes, editor, *Functional Programming Languages and Computer Architectures*, volume 523 of *Lecture Notes in Computer Science*, pages 352–366, Harvard, Massachusetts, USA, 1991. Springer, Berlin.

[JLLM91] M.J. Maher J-L. Lassez and K. Marriot. Elimination of negation in term algebras. In A. Tarlecki, editor, *Mathematical Foundations of Computer Science*, pages 1–16, Berlin, jul 1991. Springer-Verlag.

[Kah87] Gilles Kahn. Natural semantics. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, pages 22–39. Springer-Verlag LNCS 247, 1987.

[KL87] Claude Kirchner and Pierre Lescanne. Solving disequations. In *Proceedings, Symposium on Logic in Computer Science*, pages 347–352, Ithaca, New York, 22–25 June 1987. The Computer Society of the IEEE.

[KNW93] Keehand Kwon, Gopalan Nadathur, and Debra Sue Wilson. Implementing a notion of modules in the logic programming language λProlog. In E. Lamma and P. Mello, editors, *Proceedings of the Third International Workshop on Extensions of Logic Programming*, pages 359–393, Bologna, Italy, February 1993. Springer-Verlag LNAI 660.

[Kre92] Per Kreuger. GCLA II. a definitional approach to control. Technical Report R92:09, SICS, 1992.

[Lia95] Chuck Liang. *Object-Level Substitution, Unification and Generalization in Meta-Logic*. PhD thesis, University of Pennsylvania, August 1995.

[Lif99] Vladimir Lifschitz. Answer set planning. In D. De Schreye, editor, *Proceedings of the International Conference on Logic Programming (ICLP'99)*, pages 23–35. MIT Press, 1999.

[Llo93] J. W. Lloyd. *Foundations of Logic Programming, Second Extended Edition*. Springer-Verlag, 1993.

[LM87] J.-L. Lassez and K. Marriot. Explicit representation of terms defined by counter examples. *Journal of Automated Reasoning*, 3(3):301–318, September 1987.

[LMR92] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT press, Cambridge, Massachusetts, 1992.

[LT84] J. W. Lloyd and R. W. Topor. Making PROLOG more expressive. *LOGIC PROGRAM. (USA) ISSN: 0743-1066*, 1(3):225–40, October 1984.

[Lug94] D. Lugiez. Higher-order disunification: Some decidable cases. In J.-P. Jouannaud, editor, *Proceedings of the First International Conference on Constraints in Computational Logics*, pages 121–135, Munich, Germany, September 1994. Springer-Verlag LNCS 845.

[Lug95] D. Lugiez. Positive and negative results for higher-order disunification. *Journal of Symbolic Computation*, 20(4):431–470, October 1995.

[Mah88] M. Maher. Complete Axiomatizations of the Algebras of finite, infinite and rational Trees. In *Proc. third Annual Symposium on Logic in Computer Science*, pages 348–359. Computer Society Press, July 1988.

[Mak87] Johann A. Makowsky. Why Horn formulas matter in computer science: Initial structures and generic examples. *Journal of Computer and System Sciences*, 34:266–292, 1987.

[MAK91] L. Hallnass Martin Aronson, L-H. Eriksson and P. Kreuger. A survey of GCLA: a definitional approach to logic programming. In Peter Schroeder-Heister, editor, *Proceedings of the First International Workshop on Extensions of Logic Programming*, volume 1050 of *475*, pages 19–34, Berlin, March28–30 1991. Springer.

[Mal71] A.I. Mal'cev. Complete axiomatixation of classes of locally free algebrae of varioud type. In *The metamathematics of algebraic systems: Collected papers, 1936–1967*, pages 262–289. North-Holland, 1971.

[McC88a] L. Thorne McCarty. Clausal intuistionistic logic: I. Fixed point semantics. *Journal of Logic Programming*, 5(1):1–31, March 1988.

[McC88b] L. Thorne McCarty. Clausal intuitionistic logic II. Tableau proof procedure. *Journal of Logic Programming*, 5:93–132, 1988.

[Mil89a] Dale Miller. Lexical scoping as universal quantification. In G. Levi and M. Martelli, editors, *Proceedings of the Sixth International Conference on Logic Programming*, pages 268–283, Lisbon, Portugal, June 1989. MIT Press.

[Mil89b] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In Peter Schroeder-Heister, editor, *Proceedings of the International Workshop on Extensions of Logic Programming*, pages 253–281, Tübingen, Germany, 1989. Springer-Verlag LNAI 475.

[Mil89c] Dale Miller. A logical analysis of modules in logic programming. *Journal of Logic Programming*, 6(1-2):79–108, January 1989.

[Mil91] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.

[Mil92] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14:321–358, 1992.

[Mil94] Dale Miller. A multiple-conclusion meta-logic. In S. Abramsky, editor, *Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 272–281, Paris, France, July 1994.

[MM82] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, April 1982.

[MM97] Raymond McDowell and Dale Miller. A logic for reasoning with higher-order abstract syntax: An extended abstract. In Glynn Winskel, editor, *Proceedings of the Twelfth Annual Symposium on Logic in Computer Science*, pages 434–445, Warsaw, Poland, June 1997.

[MM00] Raymond McDowell and Dale Miller. Cut-elimination for a logic with definitions and induction,. *Theoretical Computer Science*, 231(231):91–119, 2000.

[MMP88] Paolo Mancarella, Simone Martini, and Dino Pedreschi. Complete logic Programs with domain-closure Axiom. *Journal of Logic Programming*, 5:263–276, 1988.

[MN89] J. Maluszyński and T. Näslund. Fail Substitutions for Negation as Failure. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings of the North American Conference on Logic Programming*, pages 461–476, Cleveland, Ohio, USA, 1989.

[MNPS91] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[Mom92] Alberto Momigliano. Minimal negation and hereditary Harrop formulae. In Anil Nerode and Mikhail Taitslin, editors, *Proceedings of Logical Foundations of Computer Science (Tver '92)*, volume 620 of *LNCS*, pages 326–335, Berlin, Germany, July 1992. Springer.

[MP88] Paolo Mancarella and Dino Pedreschi. An algebra of logic programs. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1006–1023, Seattle, 1988. ALP, IEEE, The MIT Press.

[MP91] Spiro Michaylov and Frank Pfenning. Natural semantics and some of its meta-theory in Elf. In L.-H. Eriksson, L. Hallnäs, and P. Schroeder-Heister, editors, *Proceedings of the Second International Workshop on Extensions of Logic Programming*, pages 299–344, Stockholm, Sweden, January 1991. Springer-Verlag LNAI 596.

[MP93] Spiro Michaylov and Frank Pfenning. Higher-order logic programming as constraint logic programming. In *Position Papers for the First Workshop on Principles and Practice of Constraint Programming*, pages 221–229, Newport, Rhode Island, April 1993. Brown University.

[MPRT90a] Paolo Mancarella, Dino Pedreschi, Marina Rondinelli, and Marco Tagliatti. Algebraic properties of a class of logic programs. In Saumya Debray and Manuel Hermenegildo, editors, *Proceedings of the 1990 North American Conference on Logic Programming*, pages 23–39, Austin, 1990. ALP, MIT Press.

[MPRT90b] Paolo Mancarella, Dino Pedreschi, Marina Rondinelli, and Marco Tagliatti. Algebraic properties of a class of logic programs. In Saumya Debray and Manuel Hermenegildo, editors, *Proceedings of the 1990 North American Conference on Logic Programming*, pages 23–39, Austin, 1990. ALP, MIT Press.

[Myc80] Alan Mycroft. The theory and practice of transforming call-by-need to call-by-value. In *Proceedings of the 4th International Symposium on Programming*, volume 83 of *Lecture Notes in Computer Science*, pages 269–281. Springer Verlag, 1980.

[Nai86] Lee Naish, editor. *Negation and Control in Prolog*, volume 238 of *Lecture Notes in Computer Science*. Springer Verlag, 1986.

[Nel49] D. Nelson. Constructive falsity. *Journal of Symbolic Logic*, pages 16–26, 1949.

[Nip91] Tobias Nipkow. Higher-order critical pairs. In G. Kahn, editor, *Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 342–349, Amsterdam, The Netherlands, July 1991.

[Nip93] Tobias Nipkow. Orthogonal higher-order rewrite systems are confluent. In M. Bezem and J.F. Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, pages 306–317, Utrecht, The Netherlands, May 1993.

[NL92] Olivetti N. and Terracini L. N-Prolog and the equivalence of logic program. *Journal of Logic Language and Information*, 5:253-3392, 1992.

[NL95] Gopalan Nadathur and Donald W. Loveland. Uniform proofs and disjunctive logic programming. In D. Kozen, editor, *Proceedings of the Tenth Annual Symposium on Logic in Computer Science*, pages 148–155, San Diego, California, June 1995. IEEE Computer Society Press. Available as Technical Report CS-1994-40, Department of Computer Science, Duke University, December 1994.

[NM88] Gopalan Nadathur and Dale Miller. An overview of λProlog. In Kenneth A. Bowen and Robert A. Kowalski, editors, *Fifth International Logic Programming Conference*, pages 810–827, Seattle, Washington, August 1988. MIT Press.

[O'K85] R. A. O'Keefe. Towards an algebra for constructing logic programs. In *Proceedings of the International Symposium on Logic Programming*, pages 152–161. IEEE Computer Society, Technical Committee on Computer Languages, The Computer Society Press, July 1985.

[PE88] Frank Pfenning and Conal Elliott. Higher-order abstract syntax. In *Proceedings of the ACM SIGPLAN '88 Symposium on Language Design and Implementation*, pages 199–208, Atlanta, Georgia, June 1988.

[Pea90] David Pearce. Reasoning with negative information, II: Hard negation, strong negation and logic programs. In H. Pearce, D.; Wansing, editor, *Proceedings of the International Workshop on Nonclassical Logics and Information Processing*, volume 619 of *LNAI*, pages 63–79, Berlin, FRG, November 1990. Springer Verlag.

[Pet81] G. L. Peterson. Myths about the mutual exclusion problem. *Information Processing Letters*, 12(3):115–116, June 1981.

[Pfe89] Frank Pfenning. Elf: A language for logic definition and verified meta-programming. In *Fourth Annual Symposium on Logic in Computer Science*, pages 313–322, Pacific Grove, California, June 1989. IEEE Computer Society Press.

[Pfe91a] Frank Pfenning. Logic programming in the LF logical framework. In Gérard Huet and Gordon Plotkin, editors, *Logical Frameworks*, pages 149–181. Cambridge University Press, 1991.

[Pfe91b] Frank Pfenning. Unification and anti-unification in the Calculus of Constructions. In *Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 74–85, Amsterdam, The Netherlands, July 1991.

[Pfe92] Frank Pfenning. Computation and deduction. Unpublished lecture notes, 277 pp. Revised May 1994, April 1996, May 1992.

[Pfe93] Frank Pfenning. Refinement types for logical frameworks. In Herman Geuvers, editor, *Informal Proceedings of the Workshop on Types for Proofs and Programs*, pages 285–299, Nijmegen, The Netherlands, May 1993.

[Pfe99] Frank Pfenning. Logical frameworks. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers, 1999. In preparation.

[PG86] David L. Poole and Randy Goebel. Gracefully adding negation and disjunction to Prolog. In Ehud Shapiro, editor, *Proceedings of the Third International Conference on Logic Programming*, Lecture Notes in Computer Science, pages 635–641, London, 1986. Springer-Verlag.

[Plo71] G. D. Plotkin. A further note on inductive generalization. In *Machine Intelligence*, volume 6, pages 101–124. Edinburgh University Press, 1971.

[PM90] Remo Pareschi and Dale Miller. Extending definite clause grammars with scoping constructs. In David H. D. Warren and Peter Szeredi, editors, *Proceedings of Seventh International Conference on Logic Programming*, pages 373–389, Jerusalem, Israel, June 1990. MIT Press.

[Pym99] David J. Pym. On bunched predicate logic. In G. Longo, editor, *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS'99)*, pages 183–192, Trento, Italy, July 1999. IEEE Computer Society Press.

[Rei78] R. Reiter. On closed world databases. In Gallaire and Minker, editors, *Logic and Databases*, pages 55–76. Plenum Press, New York, 1978.

[RL92] David W. Reed and Donald W. Loveland. A comparison of three Prolog extensions. *Journal of Logic Programming*, 12(1-2):25–50, January 1992.

[Rob65] J. Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[RP96] Ekkehard Rohwedder and Frank Pfenning. Mode and termination checking for higher-order logic programs. In Hanne Riis Nielson, editor, *Proceedings of the European Symposium on Programming*, pages 296–310, Linköping, Sweden, April 1996. Springer-Verlag LNCS 1058.

[Sch00] Carsten Schürmann. *Automating the meta theory of deductive systems.* PhD thesis, Carnegie Mellon University, 2000.

[SD94] Danny De Schreye and Stefaan Decorte. Termination of logic programs: the never-ending story. *The Journal of Logic Programming*, 19 & 20:199–260, May 1994.

[SH93] Peter Schroeder-Heister. Rules of definitional reflection. In M. Vardi, editor, *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 222–232, Montreal, Canada, June 1993.

[She85] John C. Shepherdson. Negation as failure II. *The Journal of Logic Programming*, 2(3):185–202, Oktober 1985.

[She88] J. C. Shepherdson. Negation in logic progamming. In J. Minker, editor, *Found. of Deductive Databases and Logic Programming*, page 19. Morgan Kaufmann, San Mateo, CA, 1988.

[She89] J. C. Shepherdson. A sound and complete semantics for a version of negation as failure. *Theoretical Computer Science*, 65(3):343–371, July 1989.

[Sny91] Wayne Snyder. *A proof theory for general unification.* Birkhauser, Boston, 1991.

[SP98] Carsten Schürmann and Frank Pfenning. Automated theorem proving in a simple meta-logic for LF. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction (CADE-15)*, pages 286–300, Lindau, Germany, July 1998. Springer-Verlag LNCS 1421.

[ST84] T. Sato and H. Tamaki. Transformational logic program synthesis. In *International Conference on Fifth Generation Computer Systems*, 1984.

[Stä92] R. F. Stärk. *The Proof Theory of Logic Programs with Negation.* PhD thesis, University of Berne, Switzerland, 1992.

[Sti88] Mark E. Stickel. A Prolog Technology Theorem Prover: implementation by an extended Prolog compiler. *Journal of Automated Reasoning*, 4:353–380, 1988.

[Stu95] Peter J. Stuckey. Negation and constraint logic programming. *Information and Computation*, 118(1):12–33, April 1995.

[SW92] D. T. Sannella and L. A. Wallen. A calculus for the construction of modular Prolog programs. *Journal of Logic Programming*, 12(1-2):147–177, January 1992.

[Thi84] Jean Jacques Thiel. Stop losing sleep over incomplete data type specifications. In Ken Kennedy, editor, *Conference Record of the 11th Annual ACM Symposium on Principles of Programming Languages*, pages 76–82, Salt Lake City, UT, jan 1984. ACM Press.

[TMM89] Kuo Tsung-Min and Prateek Mishra. Strictness Analysis: A New Perspective Based on Type Inference. In *FPCA '89, Functional Programming Languages and Computer Architecture*, London, UK, September 11–13, 1989. ACM Press, New York.

[TS84] H. Tamaki and T. Sato. Unfold/fold transformation of logic programs. In Sten-Åke Tärnlund, editor, *Proceedings of the Second International Conference on Logic Programming*, pages 127–138, Uppsala, 1984.

[Vir99] Roberto Virga. *Higher-Order Rewriting with Dependent Types.* PhD thesis, Department of Mathematical Sciences, Carnegie Mellon University, 1999. Forthcoming.

[Wad90] P. Wadler. Is there a use for linear logic. Technical report, University of Glasgow, 1990.

[Wal87] M. Wallace. Negation by constraints: A sound and efficient implementation of negation in deductive databases. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 253–263, San Francisco, August - September 1987. IEEE, Computer Society Press.

[Wan93] Heinrich Wansing. *The logic of information structures*, volume 681 of *Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science.* Springer-Verlag Inc., New York, NY, USA, 1993. Revision of the author's doctoral thesis (Fachbereich Philosophie und Sozialwissenschaften I of the Free University of Berlin).

[WBF93] D. A. Wright and C. A. Baker-Finch. Usage analysis with natural reduction types. In *Lecture Notes in Computer Science*, volume 724, pages 254–266, 1993.

[WM91] Larry Wos and William McCune. Automated theorem proving and logic programming: A natural symbiosis. *Journal of Logic Programming*, 11(1):1–53, July 1991.

[Wri91] D. A. Wright. A new technique for strictness analysis. In *TAPSOFT '91.* Springer-Verlag, New York, NY, 1991. Lecture Notes in Computer Science 494.

[Wri92] D. A. Wright. *Reduction types and intensionality in the lambda-calculus.* PhD thesis, University of Tasmania, September 1992.

[Wri96] David A. Wright. Linear, strictness and usage logics. In Michael E. Houle and Peter Eades, editors, *Proceedings of Conference on Computing: The Australian Theory Symposium*, pages 73–80, Townsville, January 29–30 1996. Australian Computer Science Communications.