NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

PhD THESIS

Semantics of Negation in Extensional Higher-Order Logic Programming

Ioanna A. Symeonidou

ATHENS

JUNE 2018

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Σημασιολογία της Άρνησης στον Εκτατικό Λογικό Προγραμματισμό Ανώτερης Τάξης

Ιωάννα Α. Συμεωνίδου

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2018

PhD THESIS

Semantics of Negation in Extensional Higher-Order Logic Programming

Ioanna A. Symeonidou

SUPERVISOR: Panagiotis Rondogiannis, Professor NKUA

THREE-MEMBER ADVISORY COMMITTEE:

Panagiotis Rondogiannis, Professor NKUA

Panagiotis Stamatopoulos, Assistant Professor NKUA

Christos Nomikos, Assistant Professor Univ. of Ioannina

SEVEN-MEMBER EXAMINATION COMMITTEE

Panagiotis Rondogiannis,
Professor NKUA

Panagiotis Stamatopoulos,
Assistant Professor NKUA

Christos Nomikos,
Assistant Professor Univ. of Ioannina

Manolis Gergatsoulis,
Professor Ionian University

Stavros Kolliopoulos,
Professor NKUA

Nikolaos Papaspyrou,
Associate Professor NTUA

Yannis Smaragdakis,
Professor NKUA

Examination Date: June 28, 2018

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Σημασιολογία της Άρνησης στον Εκτατικό Λογικό Προγραμματισμό Ανώτερης Τάξης

Ιωάννα Α. Συμεωνίδου

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Παναγιώτης Ροντογιάννης, Καθηγητής ΕΚΠΑ

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:

Παναγιώτης Ροντογιάννης, Καθηγητής ΕΚΠΑ

Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής ΕΚΠΑ

Χρήστος Νομικός, Επίκουρος Καθηγητής Παν. Ιωαννίνων

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Παναγιώτης Ροντογιάννης,
Καθηγητής ΕΚΠΑ

Παναγιώτης Σταματόπουλος,
Επίκουρος Καθηγητής ΕΚΠΑ

Χρήστος Νομικός,
Επίκουρος Καθηγητής Παν. Ιωαννίνων

Εμμανουήλ Γεργατσούλης,
Καθηγητής Ιονίου Πανεπ.

Σταύρος Κολλιόπουλος,
Καθηγητής ΕΚΠΑ

Νικόλαος Παπασπύρου,
Αναπληρωτής Καθηγ. ΕΜΠ

Ιωάννης Σμαραγδάκης,
Καθηγητής ΕΚΠΑ

Ημερομηνία Εξέτασης: 28 Ιουνίου 2018

ABSTRACT

We consider the two existing extensional approaches to the semantics of positive higher-order logic programming, originally introduced by W. W. Wadge and M. Bezem respectively. The former approach uses classical domain-theoretic tools while the latter builds on a fixed-point construction defined on a syntactic instantiation of the source program. The relationships between these two approaches had not been investigated until now, while only Wadge's approach had been extended to apply to higher-order programs with negation.

We show that Wadge's semantics and Bezem's semantics coincide for a broad and interesting class of programs, which do not include existentially quantified predicate variables in the bodies of clauses. We indicate that they also have profound differences, which surface when we extend our source language to allow existential predicate variables.

In addition, we focus on the less developed research direction of the two, namely Bezem's semantics, and we adapt, for the first time, Bezem's technique to define an extensional semantics for higher-order logic programs with negation. For this purpose, we utilize the infinite-valued approach to negation-as-failure. On the other hand, we show that an adaptation of the technique under the well-founded or the stable model semantics does not in general lead to an extensional semantics. We analyse the reasons for this failure arguing that a three-valued setting cannot distinguish between certain predicates that appear to have a different behaviour inside a program context, but which happen to be identical as three-valued relations.

As an application of our developments, we define for the first time the notions of stratification and local stratification for higher-order logic programs with negation. We prove that every stratified program has a distinguished extensional model which can be equivalently obtained through the well-founded, stable or infinite-valued model semantics. Furthermore, we show that this model does not assign the unknown truth value. These results affirm the importance and the well-behaved nature of stratified programs, which was, until now, only known for the first-order case.

SUBJECT AREA: Programming Languages

KEYWORDS: Higher-order logic programming, Negation in logic programming, Extensional semantics

ΠΕΡΙΛΗΨΗ

Θεωρούμε τις δύο υπάρχουσες εκτατικές προσεγγίσεις στη σημασιολογία των θετικών λογικών προγραμμάτων ανώτερης τάξης, προταθείσες από τον W. W. Wadge και τον M. Bezem αντίστοιχα. Η πρώτη προσέγγιση χρησιμοποιεί κλασικά εργαλεία από τη θεωρία πεδίων ενώ η δεύτερη στηρίζεται στις συντακτικές οντότητες που εμφανίζονται στο πρόγραμμα και βασίζεται στην επεξεργασία του βασικού αναπτύγματος του προγράμματος. Οι σχέσεις μεταξύ των δύο προσεγγίσεων δεν είχαν ως τώρα διερευνηθεί, ενώ μόνο η προσέγγιση του Wadge είχε επεκταθεί ώστε να εφαρμοστεί σε προγράμματα ανώτερης τάξης με άρνηση.

Δείχνουμε ότι οι σημασιολογίες του Wadge και του Bezem συμπίπτουν για μία ευρεία και ενδιαφέρουσα κλάση προγραμμάτων, τα οποία δεν περιλαμβάνουν υπαρξιακά ποσοτικοποιημένες μεταβλητές στα σώματα των προτάσεων. Σημειώνουμε ότι έχουν επίσης ουσιαστικές διαφορές, οι οποίες γίνονται εμφανείς όταν επεκτείνουμε την θεωρούμενη γλώσσα ώστε να επιτρέπονται υπαρξιακές μεταβλητές.

Επιπλέον, εστιάζουμε στη λιγότερο ανεπτυγμένη ερευνητική κατεύθυνση εκ των δύο, δηλαδή τη σημασιολογία του Bezem, και προσαρμόζουμε για πρώτη φορά την τεχνική του Bezem ώστε να ορίσουμε μία εκτατική σημασιολογία για λογικά προγράμματα ανώτερης τάξης με άρνηση. Για τον σκοπό αυτό, αξιοποιούμε την απειρότιμη προσέγγιση στην άρνηση-μέσω-αποτυχίας. Από την άλλη, δείχνουμε ότι ο συνδυασμός της τεχνικής με τη σημασιολογία σταθερού μοντέλου ή με την καλώς θεμελιωμένη σημασιολογία, αποτυγχάνει να παράξει εκτατικές σημασιολογίες, στη γενική περίπτωση. Αναλύουμε τις αιτίες αυτής της αποτυχίας και ισχυριζόμαστε ότι μία τρίτιμη λογική δεν μπορεί να διαχωρίσει μεταξύ τους ορισμένα κατηγορήματα, τα οποία έχουν διαφορετική συμπεριφορά μέσα σε ένα πρόγραμμα, αλλά τυγχάνει να εμφανίζονται ως πανομοιότυπες τρίτιμες σχέσεις.

Τέλος, ορίζουμε για πρώτη φορά τις έννοιες της στρωματοποίησης και της τοπικής στρωματοποίησης για λογικά προγράμματα ανώτερης τάξης με άρνηση. Αποδεικνύουμε ότι κάθε στρωματοποιημένο πρόγραμμα έχει ένα διακριτό εκτατικό μοντέλο, το οποίο μπορεί να κατασκευαστεί ισοδύναμα μέσω της καλώς θεμελιωμένης, της σταθερής ή της απειρότιμης σημασιολογίας. Επιπλέον, δείχνουμε ότι αυτό το μοντέλο δεν αποδίδει ποτέ την άγνωστη τιμή αληθείας. Τα αποτελέσματα αυτά αναδεικνύουν τη σπουδαιότητα και την καλή φύση των στρωματοποιημένων προγραμμάτων, που ήταν ως τώρα γνωστή μόνο στην περίπτωση των λογικών προγραμμάτων πρώτης τάξης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Γλώσσες Προγραμματισμού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Λογικός προγραμματισμός ανώτερης τάξης, Άρνηση στον λογικό προγραμματισμό, Εκτατική σημασιολογία

Στις ανιψιές μου, Αναστασία και Χριστίνα –
τις "τεχνικές διαχείρισης άγχους" μου.

ACKNOWLEDGEMENTS

It is customary for students to thank their advisor in their dissertation. I do not know how to express that this note is far from a “customary” or conventional thanks. Panos Rondogiannis has offered me much more than guidance in my research efforts. Panos is responsible for introducing me, as a teacher during my undergraduate years, to the scientific field in which I chose to work for my Bachelor’s, Master’s and Ph.D. theses, and has supervised all of these. He acted as a trigger for me to return to and finish my Master’s degree studies when I had almost given up. He made me believe that I could pursue a Ph.D. degree while working in a full-time job, then he made it possible. Most importantly, Panos has done all of the above with immense patience, encouragement and support – especially during times of scientific or personal difficulties – and with unparalleled ethos and commitment.

I owe a special thanks to Angelos Charalambidis, not only because his dissertation defense marked the beginning of my research endeavours, but also for his invaluable insights and our exchange of ideas. Angelos’s help was especially crucial when I was labouring, at the early stages of my studies, to understand the material that would be the basis of my work. Part of this material was Angelos’s dissertation, also supervised by Panos Rondogiannis.

Studying for a Ph.D. while working has been demanding, but it has been made easier through the understanding and active support of Nikos Plessas and Olga Balafa, my superiors at the two job positions I have held during my studies. The people I am proud to call my co-workers have also played a big part in this.

My family are the people who have inspired me to learn, persist and demand the best of myself, throughout my life. I am grateful for their love and their pride in my scientific achievements, even when their efforts to make sense of these achievements were less than successful.

LIST OF PUBLICATIONS

[1] Angelos Charalambidis, Panos Rondogiannis, and Ioanna Symeonidou. Approximation fixpoint theory and the well-founded semantics of higher-order logic programs. *Theory and Practice of Logic Programming*, 18:421–437, 2018.

[2] Panos Rondogiannis and Ioanna Symeonidou. The intricacies of three-valued extensional semantics for higher-order logic programs. In *27th International Joint Conference on Artificial Intelligence (IJCAI 2018) Best Sister Conferences, Stockholm, Sweden, July 13-19, 2018, Proceedings*, pages 5344–5348, 2018.

[3] Panos Rondogiannis and Ioanna Symeonidou. Extensional Semantics for Higher-Order Logic Programs with Negation (extended version of conference paper). *Logical Methods in Computer Science*, Volume 14, Issue 2, July 2018.

[4] Panos Rondogiannis and Ioanna Symeonidou. The intricacies of three-valued extensional semantics for higher-order logic programs. *Theory and Practice of Logic Programming*, 17(5-6):974–991, 2017. (Presented at the 33rd International Conference on Logic Programming (ICLP 2017), Melbourne, Australia, August 28 - September 1 2017. **Best Paper Award**).

[5] Angelos Charalambidis, Panos Rondogiannis, and Ioanna Symeonidou. Equivalence of two fixed-point semantics for definitional higher-order logic programs (extended version of conference paper). *Theoretical Computer Science*, 668:27–42, 2017.

[6] Panos Rondogiannis and Ioanna Symeonidou. Extensional semantics for higher-order logic programs with negation. In Loizos Michael and Antonis C. Kakas, editors, *Logics in Artificial Intelligence - 15th European Conference (JELIA 2016), Larnaca, Cyprus, November 9-11, 2016, Proceedings*, volume 10021 of *Lecture Notes in Computer Science*, pages 447–462, 2016.

[7] Angelos Charalambidis, Panos Rondogiannis, and Ioanna Symeonidou. Equivalence of two fixed-point semantics for definitional higher-order logic programs. In Ralph Matthes and Matteo Mio, editors, *Proceedings 10th International Workshop on Fixed Points in Computer Science, (FICS 2015), Berlin, Germany, September 11-12, 2015*, volume 191 of *EPTCS*, pages 18–32, 2015.

ΣΥΝΟΠΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

Ο παραδοσιακός λογικός προγραμματισμός βασίζεται στην ιδέα ότι το υποσύνολο της *λογικής πρώτης τάξης* που ορίζεται από τις προτάσεις Horn, μπορεί να χρησιμοποιηθεί ως μία *δηλωτική* γλώσσα προγραμματισμού. Από την πληθώρα των επεκτάσεων του λογικού προγραμματισμού που έχουν προταθεί κατά καιρούς, η μετάβαση στον *λογικό προγραμματισμό ανώτερης τάξης (higher-order logic programming)* αποτελεί μία ιδιαιτέρως ελκυστική αλλά και αμφιλεγόμενη προοπτική. Το βασικό χαρακτηριστικό αυτού του προγραμματιστικού ιδιώματος είναι η δυνατότητα να ορίζονται κατηγορήματα που επενεργούν πάνω σε κατηγορήματα.

Για τον *συναρτησιακό προγραμματισμό*, το δεύτερο επιφανές είδος δηλωτικού προγραμματισμού, οι συναρτήσεις ανώτερης τάξης είναι το καθοριστικό του χαρακτηριστικό. Αντιθετα, ο λογικός προγραμματισμός ανώτερης τάξης έχει χαρακτηριστεί ως υπερβολικά περίπλοκος και όχι αναλόγως απαραίτητος. Για παράδειγμα, ο Warren [32] ισχυρίζεται ότι οι δυνατότητες ανώτερης τάξης δεν είναι ενδιαφέρουσες, καθώς μπορούν να προσομοιωθούν στην κλασσική Prolog. Ταυτόχρονα, τα λίγα συστήματα με γνήσιες δυνατότητες προγραμματισμού ανώτερης τάξης, που υλοποιήθηκαν σε προηγούμενες δεκαετίες (με χαρακτηριστικότερα παραδείγματα την HiLog [11] και την λProlog [24, 25]), βασίζονται σε γλώσσες που παρουσιάζουν μεγάλη σημασιολογική πολυπλοκότητα.

Παρά τον όποιο σκεπτικισμό, δεν μπορεί κανείς να αγνοήσει ότι οι προαναφερθείσες γλώσσες ανώτερης τάξης έχουν πολλά πλεονεκτήματα από πλευράς εκφραστικής δύναμης και ευκολίας χρήσης. Γενικότερα, εξάλλου, η χρήση μίας προγραμματιστικής γλώσσας για την προσομοίωση των δυνατοτήτων μίας άλλης είναι συχνά εφικτή αλλά σπάνια πρακτική. Συνεπώς, είναι φυσικό να αναρωτηθούμε αν υπάρχει τρόπος να επωφεληθούμε αυτών των πλεονεκτημάτων και ταυτόχρονα να διατηρήσουμε τις επιθυμητές σημασιολογικές ιδιότητες των κλασικού λογικού προγραμματισμού πρώτης τάξης.

Οι ερευνητικές προσπάθειες των [31, 2, 3, 21, 9, 8, 10] κινούνται προς αυτήν την κατεύθυνση, προσδιορίζοντας την *νοηματική (intensional)* φύση των παραδοσιακών γλωσσών ανώτερης τάξης, όπως η HiLog και η λProlog, ως το “προβληματικό” τους (από σημασιολογικής πλευράς) σημείο. Η νοηματική προσέγγιση σημαίνει ότι ένα κατηγόρημα δεν αντιπροσωπεύει απλά το σύνολο των ορισμάτων για τα οποία είναι αληθές. Κατηγορήματα τα οποία είναι ίσα ως σχέσεις, δεν αντιμετωπίζονται απαραίτητα ως ίσα. Για παράδειγμα, στην HiLog, δύο κατηγορήματα δεν θεωρούνται ίσα εκτός αν τα ονόματά τους είναι ίδια. Τέτοιες γλώσσες δεν έχουν σημασιολογία ελάχιστου μοντέλου. Οι εναλλακτικές προσεγγίσεις των [31, 2, 3, 21, 9, 8, 10], αντίθετα, προτείνουν *εκτατικές (extensional)* γλώσσες, όπου μπορούμε να χρησιμοποιήσουμε τυπικές συνολοθεωρητικές έννοιες για να κατανοήσουμε το νόημα των προγραμμάτων και να τα μελετήσουμε. Το πλεονέκτημά τους είναι η απλή και κομψή σημασιολογία, που στηρίζεται στις εδραιωμένες τεχνικές του προγραμματισμού πρώτης τάξης. Το κόστος, ωστόσο, είναι το σχετικά περιορισμένο συντακτικό των γλωσσών αυτών.

Μπορούμε να εντοπίσουμε δύο διακριτές ερευνητικές κατευθύνσεις για την απόδοση εκτατικών σημασιολογιών στον λογικό προγραμματισμό ανώτερης τάξης. Η πρώτη [31, 21, 9, 8, 10] έχει αναπτυχθεί με τη χρήση εργαλείων της θεωρίας πεδίων και προσομοιάζει τις τεχνικές για την απόδοση δηλωτικών σημασιολογιών σε συναρτησιακές γλώσσες. Στο [31], ο W. W. Wadge έθεσε τις βάσεις αυτής της προσέγγισης, εισάγοντας έναν απλό συντακτικό περιορισμό που διασφαλίζει την ύπαρξη ενός εκτατικού ελάχιστου μοντέλου και σκιαγραφώντας τη σχετική απόδειξη. Ερευνητικές προσπάθειες που ακολού-

θησαν [21, 9] επέκτειναν και βελτίωσαν την εργασία αυτή, χαλαρώνοντας τον αρχικό περιορισμό του Wadge και δίνοντας πλήρεις και λεπτομερείς αποδείξεις όλων των αποτελεσμάτων που προδιαγράφονται στο [31], συμπεριλαμβανομένης και μίας πλήρους αποδεικτικής διαδικασίας [9]. Από το [9] προέκυψε και η πρώτη υλοποίηση μιας εκτατικής γώσσας ανώτερης τάξης, με το όνομα HOPES. Το επόμενο, προφανές βήμα ήταν η μελέτη προγραμμάτων ανώτερης τάξης με άρνηση. Πράγματι, η πρώτη εκτατική σημασιολογία για τέτοια προγράμματα δόθηκε στο [8], όπου η δηλωτική σημασιολογία του Wadge συνδυάζεται με μία σχετικά πρόσφατη πρόταση από το πεδίο της σημασιολογίας των λογικών προγραμμάτων πρώτης τάξης με άρνηση, την επονομαζόμενη *απειρότιμη σημασιολογία (infinite-valued semantics)* [30]. Τέλος, στο [10] αναπτύσσεται μία εναλλακτική (αυτής του [8]) σημασιολογία, αυτήν τη φορά επεκτείνοντας την πιο δημοφιλή και καθιερωμένη *καλώς θεμελιωμένη σημασιολογία (well-founded semantics)* [18] των γενικών προγραμμάτων πρώτης τάξης.

Η δεύτερη εκτατική προσέγγιση, στην οποία και εστιάζουμε, προτάθηκε αρχικά από τον M. Bezem στο [2, 3] για *θετικά* λογικά προγράμματα ανώτερης τάξης. Αυτή η προσέγγιση στηρίζεται στις συντακτικές οντότητες που εμφανίζονται στο πρόγραμμα και βασίζεται στην επεξεργασία του βασικού αναπτύγματος του προγράμματος. Εξαιτίας αυτών των χαρακτηριστικών, είναι σε ένα βαθμό λιγότερο ισχυρή από την προσέγγιση του [9], αλλά την ίδια στιγμή είναι απλούστερη στην κατανόηση και μπορεί πιθανώς να οδηγήσει σε ενδιαφέρουσες τεχνικές υλοποίησης. Παρόλες τις σημαντικές διαφορές τους, οι δύο ερευνητικές κατευθύνσεις δεν είναι ασύνδετες μεταξύ τους. Δείχνουμε ότι για μία ευρεία κλάση θετικών προγραμμάτων συμπίπτουν ως προς τις λογικές τιμές που αποδίδουν σε βασικά άτομα. Ωστόσο, πριν την δουλειά που παρουσιάζεται στην παρούσα διατριβή, η ερευνητική προσπάθεια που ξεκίνησε στα [2, 3] ήταν πολύ λιγότερο ανεπτυγμένη και δεν είχαν υπάρξει αποτελέσματα σχετικά με την πιθανή εφαρμογή της τεχνικής σε προγράμματα με άρνηση. Η συνεισφορά μας όσον αφορά αυτήν την προοπτική έχει και θετικές και αρνητικές αποχρώσεις: δείχνουμε ότι, ενώ είναι εφικτό να επιτύχουμε μία εκτατική σημασιολογία για γενικά προγράμματα μέσω της τεχνικής του Bezem, αυτό δεν μπορεί να γίνει μέσω των παραδοσιακών προσεγγίσεων της άρνησης. Συγκεκριμένα, δείχνουμε ότι τόσο η *σημασιολογία σταθερού μοντέλου (stable model semantics)* [19] όσο και η καλώς θεμελιωμένη σημασιολογία [18] αποτυγχάνουν σε αυτήν την αποστολή (με την αξιοσημείωτη εξαίρεση των *στρωματοποιημένων (stratified)* προγραμμάτων) και μόνον η απειρότιμη σημασιολογία [30] επιτυγχάνει. Επιπλέον, υποστηρίζουμε ότι μία τρίτιμη εκτατική σημασιολογία δεν είναι πιθανό να επιτευχθεί μέσω οποιασδήποτε τεχνικής, παρά μόνο αν κάνουμε σχετικά αντισυμβατικές υποθέσεις ως προς την συμπεριφορά της άρνησης στα λογικά προγράμματα ανώτερης τάξης. Πιο συγκεκριμένα, η συμβολή της παρούσης διατριβής μπορεί να συνοψιστεί στα παρακάτω αποτελέσματα:

- Δείχνουμε ότι για μία ευρεία και χρήσιμη κλάση προγραμμάτων χωρίς άρνηση, συγκεκριμένα για την κλάση που ορίζεται από τον W. W. Wadge στο [31], οι προσεγγίσεις των [2, 3] και [31, 21, 9] συμπίπτουν ως προς τις λογικές τιμές που αποδίδουν σε βασικά άτομα που περιέχουν σύμβολα του προγράμματος. Διαισθητικά, αυτό σημαίνει ότι για οποιοδήποτε πρόγραμμα αυτής της κλάσης, τα σύνολα των βασικών ατόμων που είναι αληθή ταυτίζονται υπό τις δύο διαφορετικές σημασιολογικές προσεγγίσεις.
- Επεκτείνουμε την θεωρούμενη γλώσσα ώστε να επιτρέπει την εμφάνιση υπαρξιακά ποσοτικοποιημένων μεταβλητών που αντιπροσωπεύουν κατηγορήματα στα σώματα των προτάσεων του προγράμματος. Αποδεικνύουμε ότι σε αυτήν την περί-

πτωση οι δύο προσεγγίσεις δίνουν εν γένει διαφορετικά αποτελέσματα, όμως παρουσιάζουμε μία επιπλέον προϋπόθεση, υπό την οποία μπορούν να συνεχίσουν να συμπίπτουν.

- Υποδεικνύουμε ότι η προσαρμογή της τεχνικής του Bezem στην σημασιολογία σταθερού μοντέλου δεν οδηγεί, στη γενική περίπτωση, σε εκτατικά μοντέλα. Ομοίως, τεκμηριώνουμε την αποτυχία της καλώς θεμελιωμένης σημασιολογίας να παράσχει εκτατικά μοντέλα σε συνδυασμό με την τεχνική του Bezem.

- Δείχνουμε ότι συνδυάζοντας την τεχνική του [2, 3] με την απειρότιμη σημασιολογία του [30], επιτυγχάνουμε μία εκτατική σημασιολογία για λογικά προγράμματα ανώτερης τάξης με άρνηση. Η απειρότιμη σημασιολογία είναι ένας ενδιαφέρων εναλλακτικός τρόπος για την ερμηνεία της άρνησης στον λογικό προγραμματισμό, στενά συνδεδεμένος με την καλώς θεμελιωμένη σημασιολογία. Να σημειωθεί επίσης, ότι η απειρότιμη σημασιολογία ήταν η πρώτη προσέγγιση της άρνησης-μέσω-αποτυχίας που κατέστησε εφικτή την επέκταση της σημασιολογίας των [31, 21, 9] (βλ. [8]).

- Μελετάμε τους λόγους της αποτυχίας της καλώς θεμελιωμένης σημασιολογίας και το γενικότερο ερώτημα της πιθανής ύπαρξης μίας εναλλακτικής τρίτιμης εκτατικής σημασιολογίας για λογικά προγράμματα ανώτερης τάξης με άρνηση. Επιχειρηματολογούμε ότι ο περιορισμός σε μία τρίτιμη λογική φαίνεται να “απορρίπτει υπερβολικά πολλή πληροφορία” και έχει σαν αποτέλεσμα, κατηγορήματα που αναμένεται να έχουν διαφορετική συμπεριφορά, να εμφανίζονται ως πανομοιότυπες τρίτιμες σχέσεις. Υποδεικνύουμε, ότι μία τέτοια τρίτιμη εκτατική σημασιολογία απαιτεί ορισμένες μη καθιερωμένες υποθέσεις σχετικά με την συμπεριφορά της άρνησης, όπως επιβεβαιώνει το παράδειγμα του [10].

- Ορίζουμε τις έννοιες της *στρωματοποίησης* και της *τοπικής στρωματοποίησης* για λογικά προγράμματα ανώτερης τάξης με άρνηση. Αυτές οι δύο έννοιες γενικεύουν τις αντίστοιχες από τον κλασικό (πρώτης τάξης) λογικό προγραμματισμό. Τέτοιες έννοιες δεν έχουν ακόμα μελετηθεί υπό το πρίσμα της σημασιολογίας των [31, 21, 9, 8, 10]. Δείχνουμε ότι οι εκδοχές της σημασιολογίας του Bezem που προκύπτουν από τον συνδυασμό της με την σημασιολογία σταθερού μοντέλου, την καλώς θεμελιωμένη σημασιολογία και την απειρότιμη σημασιολογία, συμφωνούν στην περίπτωση των στρωματοποιημένων προγραμμάτων και δίνουν ισοδύναμα εκτατικά μοντέλα. Αυτό το αποτέλεσμα επιβεβαιώνει την σημασία και την καλή συμπεριφορά των στρωματοποιημένων προγραμμάτων, που ήταν ως τώρα γνωστή μόνο για την περίπτωση του λογικού προγραμματισμού πρώτης τάξης.

Στην συνέχεια δίνουμε μία σύντομη, διαισθητική παρουσίαση της σημασιολογικής τεχνικής του Bezem για θετικά προγράμματα και σκιαγραφούμε τις τεχνικές που εφαρμόζουμε για να φτάσουμε στα αποτελέσματα που περιγράφηκαν παραπάνω.

Δεδομένου ενός θετικού λογικού προγράμματος ανώτερης τάξης, η κεντρική ιδέα πίσω από την προσέγγιση του Bezem είναι να εργαστούμε πάνω στο βασικό του ανάπτυγμα. Το τελευταίο προκύπτει αντικαθιστώντας τις μεταβλητές με καλοσχηματισμένους όρους που μπορούν να παραχθούν χρησιμοποιώντας συντακτικές οντότητες που εμφανίζονται στο πρόγραμμα. Για παράδειγμα, ας θεωρήσουμε το παρακάτω πρόγραμμα ανώτερης

τάξης:

```
q(a).
q(b).
p(Q):-Q(a).
id(R)(X):-R(X).
```

Προκειμένου να παράξουμε το βασικό ανάπτυγμα αυτού του προγράμματος, θεωρούμε κάθε πρόταση και αντικαθιστούμε κάθε μεταβλητή αυτής της πρότασης με έναν βασικό όρο που έχει τον ίδιο τύπο με την εν λόγω μεταβλητή. Με αυτόν τον τρόπο λαμβάνουμε το εξής άπειρο πρόγραμμα:

```
q(a).
q(b).
p(q):-q(a).
id(q)(a):-q(a).
id(q)(b):-q(b).
p(id(q)):-id(q)(a).
      ...
```

Πλέον, μπορούμε να αντιμετωπίσουμε το πρόγραμμα ως ένα άπειρο προτασιακό πρόγραμμα (δηλαδή, κάθε βασικό άτομο μπορεί να ειδωθεί ως μία προτασιακή μεταβλητή). Αυτό μας επιτρέπει να χρησιμοποιήσουμε τον γνωστό από τον λογικό προγραμματισμό πρώτης τάξης τελεστή άμεσου επακόλουθου (βλ. για παράδειγμα [23]), ώστε να υπολογίσουμε το σύνολο των ατόμων που θα πρέπει να θεωρηθούν “αληθή”. Στο παράδειγμά μας, το ελάχιστο σταθερό σημείο του τελεστή θα περιέχει άτομα όπως `q(a)`, `q(b)`, `p(q)`, `id(q)(a)`, `id(q)(b)`, `p(id(q))` κ.λ.π.

Ο Bezem απέδειξε ότι το ελάχιστο μοντέλο του βασικού αναπτύγματος κάθε θετικού λογικού προγράμματος ανώτερης τάξης της γλώσσας που επεξεργάζεται στο [3] είναι εκτατικό, με μία έννοια που μπορεί να ερμηνευθεί ως ακολούθως. Στο παράδειγμά μας, τα `q` και `id(q)` είναι ίσα, καθώς και τα δύο είναι αληθή για τις ίδιες ακριβώς σταθερές, δηλαδή τις `a` και `b`. Ως εκ τούτου, αναμένουμε ότι (για παράδειγμα), αν το `p(q)` είναι αληθές, τότε το `p(id(q))` θα είναι επίσης αληθές, αφού δεν μπορεί να υπάρχει διαχωρισμός μεταξύ των `q` και `id(q)`. Αυτή η ιδιότητα ορίζεται τυπικά στα [2, 3] και αποδεικνύεται ότι ισχύει για το ελάχιστο σταθερό σημείο του τελεστή άμεσου επακόλουθου του βασικού αναπτύγματος κάθε προγράμματος που συμμορφώνεται με τον απλό συντακτικό περιορισμό που επιβάλλεται από τον Bezem.

Στις προσεγγίσεις των [31, 21, 9], το μοντέλο του προγράμματος περιγράφεται αρκετά διαφορετικά. Μέσω ενός τελεστή άμεσου επακόλουθου, σε κάθε κατηγόρημα ανατίθεται μία σχέση ή, ισοδύναμα, ένα σύνολο. Για παράδειγμα, στο κατηγόρημα πρώτης τάξης `q` του προηγούμενου παραδείγματος ανατίθεται το σύνολο των αντικειμένων για τα οποία είναι αληθές, δηλαδή το σύνολο `{a, b}`. Στο κατηγόρημα δεύτερης τάξης `p` ανατίθεται το σύνολο των *μονοτονικών σχέσεων* (και όχι κατηγορημάτων) πρώτης τάξης για τα οποία είναι αληθές, δηλαδή το σύνολο `{{a},{a, b}}`. Η σύγκριση του μοντέλου αυτού με το μοντέλο που προκύπτει από την σημασιολογία του Bezem αποκαλύπτει ότι οι δύο σημασιολογίες ταυτίζονται για μία μεγάλη κλάση προγραμμάτων, τα οποία δεν περιλαμβάνουν υπαρξιακά ποσοτικοποιημένες μεταβλητές τύπου κατηγορήματος. Το αποτέλεσμα αυτό ωστόσο παύει να ισχύει στη γενική περίπτωση, όπως αποδεικνύουμε μέσω σχετικού αντιπαραδείγματος.

Η βασική ιδέα πίσω από την επέκταση της σημασιολογίας του Bezem ώστε να εφαρμοστεί σε προγράμματα ανώτερης τάξης με άρνηση, είναι εύκολο να διατυπωθεί: δεδομένου

ενός τέτοιου προγράμματος, πρώτα παίρνουμε το βασικό του ανάπτυγμα. Το αποτέλεσμα είναι ένα (πιθανώς άπειρο) προτασιακό πρόγραμμα με άρνηση, συνεπώς μπορούμε να υπολογίσουμε το νόημά του με οποιαδήποτε από τις μεθόδους που είναι διαθέσιμες για την ερμηνεία τέτοιων προγραμμάτων. Επιλέγουμε να χρησιμοποιήσουμε την καλώς θεμελιωμένη σημασιολογία [18], την σημασιολογία σταθερού μοντέλου [19] και την απειρότιμη σημασιολογία ως πιθανούς υποψήφιους [30] και κατόπιν εξετάζουμε κατά πόσο το καλώς θεμελιωμένο μοντέλο ή το κάθε σταθερό μοντέλο ή το απειρότιμο μοντέλο είναι εκτατικό κατά την έννοια των [2, 3] (όπως περιγράφεται άτυπα παραπάνω). Δείχνουμε ότι η απάντηση στο ερώτημα αυτό είναι θετική μόνο στην περίπτωση του απειρότιμου μοντέλου. Για να αποδείξουμε ότι οι σημασιολογίες που προκύπτουν από τον συνδυασμό της μεθόδου του Bezem με την σημασιολογία σταθερού μοντέλου και την καλώς θεμελιωμένη σημασιολογία δεν είναι εκτατικές, παρουσιάζουμε παραδείγματα “προβληματικών” προγραμμάτων, δηλαδή προγραμμάτων με μη εκτατικά σταθερά μοντέλα και μη εκτατικό καλώς θεμελιωμένο μοντέλο αντίστοιχα. Από την άλλη, η απόδειξη ότι η ορισθείσα απειρότιμη σημασιολογία είναι εκτατική βασίζεται σε επαγωγή που ακολουθεί τα βήματα κατασκευής του μοντέλου και σε αποδεικτικές τεχνικές από το [3].

Όπως προειπώθηκε, αποδεικνύουμε ότι κάθε στρωματοποιημένο λογικό πρόγραμμα ανώτερης τάξης με άρνηση έχει εκτατικό καλώς θεμελιωμένο μοντέλο. Επιπλέον, αυτό το μοντέλο είναι δίτιμο και ταυτίζεται με το μοναδικό σταθερό μοντέλο ενώ είναι ισοδύναμο με το απειρότιμο μοντέλο του προγράμματος. Κατ’ αυτόν τον τρόπο, υποδεικνύουμε μία ευρεία κλάση προγραμμάτων που έχουν καλή συμπεριφορά ως προς την εκτατική σημασιολογία. Η απόδειξη βασίζεται σε έναν εκ των πολλών χαρακτηρισμών του τέλειου μοντέλου, συγκεκριμένα στον κατασκευαστικό ορισμός από το [28], και στην ισοδυναμία αυτού με τα προαναφερθέντα μοντέλα.

CONTENTS

PREFACE

The research described in this dissertation was conducted under the supervision and indispensable guidance of Professor P. Rondogiannis in the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens, between November 2014 and June 2018.

This work has been presented in a series of international conferences and journal articles, as detailed in the section titled “List of Publications”. I am honoured that the paper “The intricacies of three-valued extensional semantics for higher-order logic programs” received the *Best Paper Award* at the 33rd International Conference on Logic Programming (ICLP 2017), one of the premier conferences on logic programming.

1. INTRODUCTION

The purpose of this dissertation is to evaluate M. Bezem's extensional semantics for higher-order logic programs, first in comparison to the sole existing alternative, introduced by W. W. Wadge, and second with respect to its potential to generalise to programs with negation. We demonstrate that the two approaches are closely connected and that they even coincide for a broad class of programs. Then we extend Bezem's semantics under the well-founded, stable model and infinite-valued semantics and show that only the latter succeeds in retaining extensionality in the general case. The results of the dissertation contribute to a better understanding of the semantics of higher-order logic programming and, in particular, the behaviour of negation in higher-order logic programming.

This chapter offers an introductory presentation of the main concepts involved in defining the problems addressed in this dissertation, as well as the contributions made towards the solutions. We begin with an intuitive discussion of what higher-order logic programming is and why it is a particularly interesting extension of classical (first-order) logic programming. Then we outline the main issues related to the semantics of higher-order programs and the different approaches that have been taken in attempt to define such a semantics. We conclude with a summary of our contributions and an overview of the structure of the dissertation.

1.1 First-order Logic Programming

Traditional logic programming is based on the idea of using the Horn subset of *first-order logic* as a *declarative* programming language. In contrast to *imperative* programming languages, the building blocks of a logic program are not instructions on how to perform basic tasks in a specified order, but *clauses* that describe relationships between objects. Clauses may be unconditional (e.g., "A holds") or conditional axioms (e.g., "A holds if B and C both hold"). For example, take the following program:

```
man(socrates).
god(zeus).
mortal(X):-man(X).
```

The first line is an unconditional clause and declares that an object of our world, Socrates, has a property, i.e. is a man. In logic programming terms, the *predicate* `man` is true of the *individual* `socrates`. The second line can be understood in a similar way. The third line is a conditional clause and declares that an arbitrary object, represented by the variable `X`, is mortal, if the object is a man. Predicates like `man`, `god` and `mortal` can be seen as relations, that may be true or false of different objects in our knowledge base.

Given the above program, the system can accept queries about the knowledge base described in it. It may answer questions about specific objects, as for example:

```
?- mortal(socrates).
```

In this case, the system will find that, in order for Socrates to be proven mortal, it suffices to prove that Socrates is a man. The latter is stated to hold in our world, therefore the above question would receive a positive answer. On the other hand, queries like:

```
?- mortal(zeus).
?- mortal(plato).
```

would receive negative answers, because neither Zeus nor Plato can be proven to be men on the grounds of our example program. The system can also accept queries of the following form:

```
?- mortal(Y).
```

This question can be understood as “who in our world is mortal?”. A query such as the above will be addressed by trying to bind the variable `Y` to objects for which `mortal(Y)` can be proven; in this case, Socrates. Therefore, the answer `Y=socrates` will be produced. It becomes apparent that, with declarative languages in general and logic programming in particular, handling the operational details of how the solution to a problem is reached, falls not on the programmer, but on the system executing the program.

The semantics of first-order logic programs such as the above are given by their *minimum Herbrand model* (or simply *minimum model*), i.e. the set of facts that can be deduced to be true from the program. More generally, a model of the program is a set of facts that satisfy all the clauses of the program, if they are assumed to be true, and the minimum model is the smallest such set. For our example program above, the minimum model is the set $\{$`man(socrates), god(zeus), mortal(socrates)`$\}$, which satisfies the two unconditional clauses of our program, because the facts stated by those clauses are assumed to be true in the model, and the third, conditional clause, because `mortal` is assumed to be true for every individual of which `man` is true (i.e. `socrates`).

1.2 Higher-order Logic Programming

Out of the many extensions of traditional logic programming that have been proposed over the years, the transition to a higher-order setting has been a particularly intriguing and at the same time controversial potential course. The key characteristic of *higher-order logic programming* is that (roughly speaking) it allows predicates to be passed as parameters of other predicates. In other words, we can define relations on objects, which are considered first-order, but we can also define relations on other relations, which are considered higher-order.

Example 1. The following is a higher-order program that defines the union of two relations `P, Q` (in this chapter, we use ad-hoc Prolog-like syntax for our examples):

```
union(P,Q)(X):-P(X).
union(P,Q)(X):-Q(X).
```

The predicate `union` is second-order and operates on two first-order predicates, `P` and `Q`, and an individual, `X`. □

The greatest advantage of higher-order logic programming, compared to first-order logic programming, is of course the added expressivity. Most importantly, this is achieved in a way that enhances, rather than disrupts, the positive aspects of the declarative programming style, such as code elegance, human-friendly structure and programmer productivity.

Functional programming, the other prominent declarative paradigm, boasts higher-order functions as its most elemental feature. In [20], J. Hughes advocated the importance of higher-order functions and portrayed them as a powerful structure to reinforce modular programming, by better enabling the programmer to modularize a problem conceptually. The way in which simple functions, implementing basic tasks, can be combined together

to produce more complex ones, through higher-order capabilities, is illustrated even in seemingly trivial programs in [20]. We use a similar example taken from [31]. Consider the following first-order program for deciding whether a list of integers is ordered:

```
ordered([]).
ordered([X]).
ordered([X,Y|T]) :- X<Y, ordered([Y|T]).
```

where `[]` represents an empty list, `[X]` represents a list of one element and `[X,Y|T]` represents a list with at least two elements, `X` and `Y`, followed by a possibly empty sequence of additional elements `T`. The only part of this definition that is specific to integer values is the comparison `X<Y`, while the rest of the definition only describes the iteration through the elements of the list. However, in a first-order language, implementing a similar predicate for lists of e.g. sets or strings, would require the repetition of the iteration mechanism. The use of a higher-order predicate, on the other hand, could allow the separation of the iteration function and the comparison function. This way the former becomes independent of the type of the list items and so the code can be appropriately reused if needed. Such a universal predicate might be defined as in the following example.

Example 2. The following is a higher-order, parametrized version of the above predicate `ordered`:

```
ordered([],R).
ordered([X],R).
ordered([X,Y|T],R) :- R(X,Y), ordered([Y|T],R).
```

where `R` can be substituted for any (first-order) ordering relation, such as $<$, $\subset$, a relation that checks lexicographical order, and so on. □

1.3 Issues with Higher-order Logic Programming

Programming in higher-order dialects has become a growing trend in the recent years, as the benefits of higher-order structures have become widely acknowledged. This is evident from the increasing popularity of purely functional languages, as well as the fact that higher-order features are being gradually incorporated into mainstream, traditionally imperative languages such as C# and Java.

On the opposite side, higher-order logic programming has been met with relative scepticism and has sometimes been argued to be more complicated than it is necessary. For example, D. Warren claimed in [32] that higher-order features in logic programming are of no interest, because they can be simulated in classical Prolog. At the same time, the few logic programming systems with genuinely higher-order capabilities that were implemented a while ago (most notably HiLog [11] and λProlog [24, 25]), are based on languages which suffer from semantic complexity. In the general case, arbitrary higher-order logic programs cannot be interpreted in the same way as first-order logic programs; even the existence of a minimum model is not guaranteed.

Despite the issues raised, one cannot disregard the fact that the aforementioned higher-order languages have much to offer in terms of expressive power and ease of use. After all, in programming, using one language to emulate another is very often possible, but it is rarely practical. It is natural to wonder if one can enjoy some of these appealing qualities and at the same time retain the elegant and desirable semantic properties of classical first-order logic programming.

As mentioned previously, in classical first-order logic programming, predicates denote relations and they can be identified with the set of arguments for which they are true. The semantics of these programs is said to be *extensional*. The following example from [3], based on a similar example from [31], illustrates the problems encountered when we attempt to interpret a higher-order program in an extensional way.

Example 3. Consider the program

```
p(a).
q(a).
r(p).
p(b):-r(q).
```

If we see predicates as denoting relations in the set theoretic sense, then we should see `p` and `q` as both denoting the set $\{a\}$. Since the two predicates are equal as sets, they should be indistinguishable and, therefore, interchangeable. This means that `r(p)` being true, makes it necessary that `r(q)` is also true. Then, so that the last clause is satisfied, `p` must be true of `b`. Observe that this constitutes a paradox, because adding `b` to the denotation of `p` makes `q` no longer equal to `p` and there is no more reason to assume that `r(q)` is true.

The above program does have a minimum model, namely $\{p(a), q(a), r(p)\}$, but this model is not extensional. On the other hand, it has two minimal extensional models: {`p(a), p(b), q(a), r(p)`} and {`p(a), q(a), q(b), r(p)`}, but `p(b)` in the first model and `q(b)` in the second have no justification. Moreover, in the first model `r` is only true of the relation $\{a, b\}$, while in the second it is true of a different relation, $\{a\}$. These symptoms are inconsistent with our experience with the semantics of logic programs. □

One way to address this problem is to abandon extensionality. In the traditional higher-order languages such as HiLog and λProlog mentioned above, an *intensional* approach is taken instead; predicates that are equal as relations, may not always be treated as equal. For example, in Hilog, two predicates are not considered equal unless their names are the same. In an intensional system, it is acceptable that `r(p)` succeeds and `r(q)` fails. Languages like these have all the expressive power of higher-order Horn logic. However, this choice also means abandoning the elegant, well-established semantic properties of first-order logic programs and stirring logic programming away from its logical foundation.

Extensionality, on the other hand, bears not only the theoretical advantage, but also some more practical rewards. First and foremost, the fact that two predicates which are true of the same arguments are semantically indistinguishable, has major impact on the modularity of programs. For example, if we define two sorting predicates `merge_sort` and `quick_sort` that perform the same task (possibly with different efficiency), it is *guaranteed* that any higher-order predicate will have the same behaviour whether it is given `merge_sort` or `quick_sort` as an argument. This means we can safely replace one implementation of sort with another without breaking our program. In the previous section we argued that this property is one of the major assets of functional programming; note that functional programs also have extensional semantics. As mentioned in [31] “extensionality means exactly that predicates are used as *black boxes* - and the “black box” concept is central to all kinds of engineering”. Another impressive result of extensionality is demonstrated in Example 6.

The above discussion suggests that the problem revealed in our example program deserves some more consideration. Recall that the paradox arose when we claimed that `p` and `q` being equal as sets means that if `r(p)` is true, then `r(q)` should also be true. There is one more issue with this clause: the only way to deduce from this program that `r(q)` is true, is to prove that `p` and `q` are equal. In this case this was trivial, but in general higher-order predicates may denote relations that are infinite. Therefore relying on such comparisons is unrealistic. Obviously, the root of the problem is the fact that we explicitly named a specific predicate, i.e. `p`, as a member of a certain relation, i.e. the extension of `r`. In [31] W. W. Wadge suggested that if we appropriately restrict the syntax of higher-order logic programming to disallow this practice, then we can obtain languages that can be assigned an extensional semantics. The same idea was stated independently by M. Bezem in [2], even though it was pursued through different techniques.

1.4 Syntactic Restrictions for Extensional Higher-Order Logic Programs

W. W. Wadge was the first to acknowledge that an extensional approach to higher-order logic programming would require to sacrifice some of the expressive power of higher-order logic. The syntactic restrictions imposed by Wadge in [31] are the following:

R_1**:** In the head of every clause in a program, each argument of predicate type must be a variable; all such variables must be distinct.

R_2**:** The only variables of predicate type that can appear in the body of a clause, are variables that appear in its head (in other words, existentially quantified predicate variables are not allowed).

Programs that satisfy the above restrictions are named *definitional* in [31].

Example 4. The programs of Example 1 and Example 2 (as well as the two first-order programs of Sections 1.1 and 1.2) are legitimate under Wadge's restrictions. However, the program of Example 3 does not satisfy R_1: recall that the predicate constant `p` appears as an argument in the head of the clause `r(p).` Similarly, the program:

```
p(Q,Q):-Q(a).
```

is problematic because the predicate variable `Q` is used twice in the head of the clause. Finally, the program:

```
p(a):-Q(a).
```

is not definitional because the predicate variable `Q` that appears in the body of the above clause, does not appear in the head of the clause. □

In [9], Charambidis et al. rendered unnecessary the second restriction, R_2, showing that existential predicate variables can freely appear in the bodies of clauses (and, obviously, program goals), without compromising extensionality. Example 6 demonstrates the added expressive power of dropping this restriction. However, the first restriction, R_1, is essential for the reasons discussed in Section 1.3 and so it was adopted in [9].

Interestingly, M. Bezem independently followed a very similar course. In [2] he imposed his own syntactic restrictions that guaranteed extensionality of higher-order logic programs and these were very similar to Wadge's original ones (R_1 and R_2). In particular, Bezem required that:

R_1'**:** In every atom of every clause in a program, each argument of predicate type must be a variable; all such variables in the head of the clause must be distinct.

R_2'**:** The only variables of predicate type that can appear in the body of a clause, are variables that appear in its head, or variables of first-order predicate types.

Observe that Bezem's first rule, R_1', is stricter than Wadge's respective rule, since it forbids predicate constants to appear as arguments *anywhere* in a clause (as opposed to the head atom). On the other hand, R_2' is less strict than R_2, because it declares that existentially quantified predicate variables are allowed, if they are first-order.

However, Bezem soon revisited his work and in [3] he arrived at the same conclusion as Charalambidis et al. [9] would reach a few years later: Wadge's first syntactic restriction, R_1, alone is enough to ensure that any conforming program can be assigned an extensional semantics.

It is perhaps worth noting that, despite agreeing on R_1, the language considered in [3] is still a strict superset of the language of [9]. The difference is that Bezem allows the left-most predicate of the head atom of a clause to be a variable. For example, the clause:

```
Q(b):-r(Q).
```

is legitimate under Bezem's syntax of [3], but not under the syntax of Wadge or Charambidis et al. We will not consider such programs (named *hoapata* in [3]).

1.5 Intuitive Overview of the Existing Approaches

Two distinct research directions for providing extensional semantics to higher-order logic programs can be identified. The first [31, 21, 9, 8, 10] has been developed using domain-theoretic tools, and resembles the techniques for assigning denotational semantics to functional languages. In [31] W. W. Wadge laid the foundations for this approach by introducing the syntactic restrictions discussed in Section 1.4 and outlining an extensional semantics for *positive* (i.e., negationless) higher-order logic programs. Later research efforts [21, 9] extended and refined the work of [31], relaxing Wadges's original syntactic restriction and providing complete and detailed proofs of all results either sketched or conjectured in [31], including a sound and complete proof procedure [9]. The first actual implementation of an extensional higher-order language, called HOPES, was built based on this work [9].

The second extensional approach, and the one we mainly focus on, was initially proposed by M. Bezem in [2, 3], also for positive programs. This approach relies on the syntactic entities that exist in a program, and is based on processing the *ground instantiation* of the program (the notion of ground instantiation is informally explained later in this section and formally defined in Chapter 4). Because of these traits, it is a little less powerful than the approach of [9], but at the same time, it is easy to comprehend and can probably lead to interesting implementation techniques. Despite their important differences, the two research directions are not unrelated. We show that for a broad class of positive programs, they coincide with respect to ground atoms.

We give an intuitive overview of each of the two approaches, starting with the domain theoretic semantics of [31]. The formal details of both approaches will be given in Chapter 4.

As it is argued in [31], if a program is definitional, i.e. satisfies the two syntactic restrictions, R_1 and R_2, presented in Section 1.4, then it has a *unique minimum model*.

Example 5. Consider the definitional program:

```
q(a).
q(b).
p(Q):-Q(a).
id(R)(X):-R(X).
```

In the minimum model of the above program, the meaning of `q` is the relation $\{\mathtt{a}, \mathtt{b}\}$. The meaning of `p` is the set of all unary relations (over the set of individuals appearing in the program), that contain at least `a`; more formally, it is the relation $\{r \mid \mathtt{a} \in r\}$. The meaning of `id` is the set of all pairs (r, d) such that d belongs to r; more formally, it is the relation $\{(r, d) \mid d \in r\}$. □

As remarked by W. W. Wadge (and formally demonstrated in [21, 9]), the minimum model of every definitional program is monotonic and continuous.[1] Intuitively, monotonicity means that if in the minimum model the meaning of a predicate is true of a relation, then it is also true of every superset of this relation. For example, we see that since the meaning of `p` is true of $\{\mathtt{a}\}$, then it is also true of $\{\mathtt{a}, \mathtt{b}\}$ (because $\{\mathtt{a}, \mathtt{b}\}$ is a superset of $\{\mathtt{a}\}$).

The minimum model of a given definitional program can be constructed as the least fixed-point of an operator that is associated with the program, called the *immediate consequence operator* of the program. As is demonstrated in [31, 21], the immediate consequence operator is monotonic, and this guarantees the existence of the least fixed-point which is constructed by a bottom-up iterative procedure.

Reasoning with predicates as abstract relations, as opposed to predicates as concrete syntactic entities, makes possible a powerful feature of the language HOPES. It is the fact that abstract relations can be produced by the system as answers to queries, even when these queries are not satisfied by specific relations defined in a program. The following example from [9] demonstrates this.

Example 6. Suppose that we have the following database of musicians:

```
singer(sally).
singer(steve).
drummer(dave).
quitarist(george).
quitarist(grace).
```

Then the following clause can be used to describe what constitutes a band:

```
band(B):-singer(S),B(S),drummer(D),B(D),guitarist(G),B(G).
```

This says that a band is a group that has at least a singer, a drummer and a guitarist. The programming language HOPES allows the query `?- band(B).`, at which it will start

[1] The notion of continuity will not play any role in this dissertation.

enumerating the *finite* sets of musicians that constitute bands according to the above rule. For example, answers such as $B = \{\texttt{sally, dave, george, grace}\}$ and $B = \{\texttt{steve, dave, george}\}$ will be produced. For a more detailed discussion of the example and the way that HOPES handles goals with uninstantiated predicate variables, see [7]. □

It should be noted that Bezem's semantic technique [2, 3], though also extensional, falls short in this case, as it is more syntax-oriented and cannot produce abstract answers as in the above example. Specifically, under Bezem's semantics predicate variables range over the equivalence classes, with respect to extensional equality, of predicates that appear in the program, instead of sets of abstract relations. The repercussions of this key difference between the approaches of [2, 3] and [31, 21, 9] will be discussed in Section 4.4.2. For the moment, we give a short description of Bezem's technique at an intuitive level.

Given a positive higher-order logic program, the starting idea behind Bezem's approach is to take its "ground instantiation", in which we replace variables with well-typed terms that can be created using syntactic entities that appear in the program. For example, consider the higher-order program below:

```
q(a).
q(b).
p(Q):-Q(a).
id(R)(X):-R(X).
```

In order to obtain the ground instantiation of this program, we consider each clause and replace each variable of the clause with a ground term that has the same type as the variable under consideration (the formal definition of this procedure will be given in Definition 38). In this way we obtain the following infinite program:

```
q(a).
q(b).
p(q):-q(a).
id(q)(a):-q(a).
id(q)(b):-q(b).
p(id(q)):-id(q)(a).
      ...
```

One can now treat the new program as an infinite propositional one (i.e., each ground atom can be seen as a propositional variable). This implies that we can use the standard minimum model of classical logic programming in order to obtain the set of atoms that should be taken as "true". In our example, the minimum model will contain atoms such as `q(a)`, `q(b)`, `p(q)`, `id(q)(a)`, `id(q)(b)`, `p(id(q))`, and so on.

Bezem demonstrated that the minimum model semantics of the ground instantiation of every positive higher-order logic program of the language considered in [3], is extensional in a sense that can be explained as follows. In our example, `q` and `id(q)` are equal since they are both true of exactly the constants `a` and `b`. Therefore, we expect that (for example) if `p(q)` is true then `p(id(q))` is also true, because `q` and `id(q)` should be considered as indistinguishable. This property of "indistinguishability" is formally defined in [2, 3] and it is demonstrated that it holds in the minimum model of the ground instantiation of every program that abides to the simple syntactic restriction given in Section 1.4.

The obvious next step of course is to consider higher-order programs with negation. Indeed, the first extensional semantics for such programs was given in [8], where Wadge's denotational semantics is combined with a relatively recent semantic proposal from the sphere of first-order logic programming with negation, called the *infinite-valued* semantics [30]. In [10] an alternative semantics was obtained, this time by extending the more popular and established well-founded semantics [18] of general first-order programs. However, prior to the work reported in this dissertation, the line of research started in [2, 3] was much less developed and there had been no results regarding the possible application of the technique to general logic programs.

1.6 Contributions

In this dissertation we focus exclusively on the extensional approaches to the semantics of higher-order logic programming. Our first goal is to explore the connections between the two existing research directions for positive (i.e., negationless) higher-order logic programs, namely M. Bezem's approach of [2, 3] and the one of [31, 9], initiated by W. W. Wadge and extended by Charalambidis et al. Our second and main goal, is to fill a gap in the field of extensional semantics for higher-order logic programs with negation, by investigating the applicability of Bezem's semantic technique to such programs.

The key idea behind extending Bezem's semantics in order to apply to programs with negation, is straightforward to state: given such a program, we first take its ground instantiation. The resulting program is a (possibly infinite) propositional program with negation and therefore we can compute its semantics in any standard way that exists for obtaining the meaning of such programs. We use the well-founded semantics [18], the stable model semantics [19], and the infinite-valued semantics [30] as possible candidates and then proceed to examine whether the well-founded model or each stable model or the minimum infinite-valued model is extensional in the sense of [2, 3] (informally described in the previous section).

Our contributions can be summarized as follows:

- We demonstrate that for a very broad class of *positive programs*, namely the class of definitional programs introduced by W. W. Wadge [31], the approaches of [2, 3] and [31, 9] coincide with respect to ground atoms that involve symbols of the program. Intuitively, this means that for any given definitional program, the sets of true ground atoms of the program are identical under the two different semantic approaches.
- On the other hand, we argue that if we consider an extended language, which allows existential predicate variables in the bodies of program clauses, then the two approaches give different results in general; even in this extended case however, we demonstrate that under an additional assumption, the two approaches can still be shown to coincide.
- We show that, even though it is possible to acquire an extensional semantics for general programs by Bezem's technique, this cannot be done through the traditional approaches to negation. We demonstrate that the stable model adaptation of Bezem's technique does not in general lead to extensional models. In particular, we exhibit a program that has more than one stable models, none of which are extensional. In the same manner, we demonstrate that the well-founded extension of Bezem's technique also *fails* to retain extensionality in the general case.

- We demonstrate that by combining the technique of [2, 3] with the infinite-valued semantics of [30], we obtain an extensional semantics for higher-order logic programs with negation. The infinite-valued approach is an interesting alternative way to interpret negation in logic programs, with close connections to the well-founded semantics. As it was recently demonstrated in [16, 6], despite these connections the infinite-valued approach satisfies all identities of iteration theories [5], while the well-founded semantics does not. Iteration theories (intuitively) provide an abstract framework for the evaluation of the merits of various semantic approaches for languages that involve recursion. Note also, that the infinite-valued semantics was the first approach to negation to enable the extension of the semantics of [31, 9] (see [8]).
- We study the reasons for the failure of the well-founded adaptation of Bezem's technique and the more general question of the possible existence of an *alternative* extensional three-valued semantics for higher-order logic programs with negation. We argue that restricting attention to three-valued logic appears to "throw away too much information" and makes predicates that are expected to have different behaviours, appear as identical three-valued relations. We indicate that in order to achieve such a semantics, one has to make some (arguably) non-standard assumptions regarding the behaviour of negation, for example as in the case of [10].
- We define the notions of *stratification* and *local stratification* for higher-order logic programs with negation. These two new notions generalize the corresponding ones from classical (first-order) logic programming. Such notions have not yet been studied under the semantics of [31, 9, 8, 10]. We prove that the stable model, the well-founded and the infinite-valued adaptations of Bezem's technique all agree in the case of *stratified* programs and give equivalent extensional models. This result affirms the importance and the well-behaved nature of stratified programs, which was, until now, only known for the first-order case.

1.7 Summary of the Dissertation

The rest of the dissertation is structured as follows:

In Chapter 2 we recall the syntax and semantics of *positive propositional programs*. Similarly, in Chapter 3 we discuss the syntax and semantics of propositional programs with negation. We focus on propositional programs (logic programs that do not contain variables) as these are simpler than first-order logic programs and suffice for the purposes of this dissertation. However, all the semantic approaches presented in the first two chapters also apply to, and were originally defined for, first-order logic programs.

Chapter 4 introduces higher-order logic programs and gives the formal details for the two existing approaches to their semantics. Moreover, we study the relationships between these approaches. First we prove their equivalence for definitional programs and then we discuss their differences in the case that existentially quantified predicate variables are allowed in our programs.

In Chapter 5, we consider higher-order programs with negation and give a framework for extending Bezem's semantics to such programs over two-valued, three-valued or infinite-valued interpretations. We utilize this framework in conjunction with the stable model semantics and the well-founded model semantics and show that both of these approaches fail to produce extensional semantics for our higher-order programs. The

extension under the infinite-valued semantics, on the other hand, is shown to achieve this goal. Finally, we define the classes of stratified and locally stratified programs and show that they have equivalent extensional models under all the extensions of Bezem's semantics.

Chapter 6 concludes the dissertation with a discussion of related work and possible future research directions.

2. POSITIVE PROPOSITIONAL LOGIC PROGRAMS

One of the advantages of the extensional approaches to the semantics of higher-order logic programming, is that they are usually generalisations of existing semantic techniques from traditional first-order logic programming. The purpose of this chapter and the next is to give an overview of the specific techniques for positive and general programs respectively, which we will employ.

Even though we have often referred (and will continue to refer) to the semantics of first-order programs, we will restrict our attention to *propositional* programs, for two reasons. First, M. Bezem's semantics [2, 3] and our extensions of it, as described intuitively in Section 1.5, are based on the idea of seeing the ground instantiation of a higher-order program as a propositional program. Then, the aforementioned techniques from the first-order case are applied to this perceived propositional program (and not a first-order program more generally). The second reason is simplicity: the syntax of such programs is almost elementary and the concepts involved in the definitions of the individual semantic approaches are often simplified in the absence of variables.

2.1 Syntax

In this section, we define the syntax of positive propositional logic programs.

We begin by defining our source language, $\mathcal{L}$.

Definition 1. The *alphabet* of $\mathcal{L}$ consists of:

- the set of *propositional variables* (denoted by lower case letters, such as $\mathsf{p}, \mathsf{q}, \ldots$);
- the constants *true* and *false*;
- the inverse implication constant $\leftarrow$;
- the comma.

Note that, by a common convention in logic programming, the comma is used in place of the logical conjunction constant $\wedge$.

Definition 2. A *propositional clause* (or simply *clause*) of $\mathcal{L}$ is a formula $\mathsf{p} \leftarrow \mathsf{A}_1, \ldots, \mathsf{A}_m$, where $m \geq 0$, p is a propositional variable and each A_i is either a propositional variable or one of the constants *true*, *false*. The propositional variable p is called the *head* of the clause and the conjunction $\mathsf{A}_1, \ldots, \mathsf{A}_m$ is its *body*. A *propositional program* (or simply *program*) P of $\mathcal{L}$ is a countable set of clauses.

The above definition is slightly different from the usual definition of a program in two ways. First, it allows the constants *true* and *false* to appear in the body of a clause. Second, it permits the program to comprise an infinite number of clauses. Both of these assumptions are necessary for technical reasons that will be made clear in Chapters 4 and 5. Also, they are compatible with all of the semantic approaches presented in this and the next chapter.

2.2 Minimum Model Semantics

Ascribing meaning to propositional programs is fairly straightforward. We think of propositional variables as representing atomic statements that may be true or false and we think of clauses as more complex statements, which tell us that if a set of atomic statements (i.e., the propositional variables in the body of a clause) are simultaneously true, then it is implied that a different statement (i.e., the head of the clause) is also true. This way, clauses may also be true or false, depending on the values of the atomic statements that they involve.

A *(Herbrand) interpretation* of the program is a stipulation of the truth values of the propositional variables that appear in the program. On the other hand, we regard the constants *true* and *false* as having fixed truth values across all interpretations. These concepts are described in the following two definitions:

Definition 3. The set of propositional variables that appear in a program P is called the *Herbrand base* of P and denoted by B_{P}.

Definition 4. A *Herbrand interpretation* (or simply *interpretation*) I of a program P is a mapping from B_{P} to the set $\{T, F\}$. Moreover, we define $\mathsf{I}(\textit{true}) = T$ and $\mathsf{I}(\textit{false}) = F$.

The set $\{T, F\}$ is the traditional two-valued truth domain and will be denoted by $\mathbb{V}^2$. In the next chapter, we will define more truth domains with additional truth values.

Frequently, an interpretation I is identified with a subset of the Herbrand base; in particular, with the set $\{\mathsf{p} \mid \mathsf{p} \in B_{\mathsf{P}} \text{ and } \mathsf{I}(\mathsf{p}) = T\}$. This means that we can write either $\mathsf{I}(\mathsf{p}) = T$ or $\mathsf{p} \in \mathsf{I}$ to denote that the propositional variable p is assigned the value T in I. We will make this identification throughout and use the two notations interchangeably.

We define an ordering on the set of interpretations based on the usual ordering of truth values, i.e. $F \leq F$, $T \leq T$ and $F \leq T$.

Definition 5. Let P be a program. We define the relation $\leq$ on the set of interpretations of P as follows: if I and J are interpretations of P, then $\mathsf{I} \leq \mathsf{J}$ if, for all propositional variables $\mathsf{p} \in B_{\mathsf{P}}$, we have $\mathsf{I}(\mathsf{p}) \leq \mathsf{J}(\mathsf{p})$.

As mentioned earlier, every clause of a program is intended to state that, if the propositional variables (and constants) in the body of the clause are all true with respect to an interpretation of the program, then the propositional variable that is the head of the clause is also true with respect to the same interpretation. Of course this may be so for some interpretations, but not for others. This leads to the concept of *(Herbrand) model*:

Definition 6. An interpretation M of a propositional program P is a model of P, if for every clause $\mathsf{p} \leftarrow \mathsf{A}_1, \ldots, \mathsf{A}_m$ in P, it holds that $\mathsf{M}(\mathsf{p}) \geq min\{\mathsf{M}(\mathsf{A}_1), \ldots, \mathsf{M}(\mathsf{A}_m)\}$.

As it is well established in the literature (for example [23]), every program has a unique minimum (with respect to $\leq$) model. This model can be characterized as the intersection of all Herbrand models of the program or as the least fixed-point of the so called *immediate consequence operator* of the program:

Definition 7. We define the *immediate consequence operator,* T_{P}*, for* P, as follows:

$$T_{\mathsf{P}}(\mathsf{I})(\mathsf{p}) = \begin{cases} T, & \text{if there exists a clause } \mathsf{p} \leftarrow \mathsf{A}_1, \ldots, \mathsf{A}_m \text{ in P} \\ & \text{such that } \mathsf{I}(\mathsf{A}_i) = T \text{ for all } i \in \{1, \ldots, m\} \\ F, & \text{otherwise.} \end{cases}$$

It is widely accepted that the minimum Herbrand model captures the intended meaning of positive logic programs.

3. GENERAL PROPOSITIONAL LOGIC PROGRAMS

The most obvious and intuitive way in which we can expect to extend the expressive power of logic programs is the addition of negation, most often in the bodies of clauses. Unfortunately, this seemingly natural extension disrupts the relationship between logic programs and their underlying logics. As a result, the semantics of negation proves to be a complex problem for almost every flavour of logic programming. This is made evident from the fact that numerous different approaches have been proposed even for classical first-order logic programs with negation (for example [14, 1, 17, 19, 18, 30]), some of which are discussed in the following sections.

We continue to examine possibly infinite propositional programs, which we extend to allow negated propositional variables in the bodies of clauses. As mentioned in the previous chapter, the semantic approaches presented here apply to such programs, even though they were originally defined for finite first-order programs.

3.1 Syntax

In this section we extend the syntax of propositional programs so as to allow negation. We introduce the negation operator $\sim$ which can be applied to propositional variables.

Definition 8. A *positive literal* is a propositional variable. A *negative literal* is an expression of the form $\sim$p, where p is a propositional variable. A *literal* is either a positive or negative literal.

Our extended class of logic programs is defined by the fact that the body of a clause may contain both positive and negative literals:

Definition 9. A *general propositional clause* of $\mathcal{L}$ is a formula $\mathsf{p} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_m$, where $m \geq 0$, p is a propositional variable and each L_i is either a literal or one of the constants *true*, *false*. A *general propositional program* P of $\mathcal{L}$ is a countable set of clauses.

A general propositional program (respectively, general propositional clause) may also be called a propositional program or simply a program (respectively, propositional clause or simply clause), when no confusion arises. On the other hand, when we need to refer to programs (respectively, clauses) that do not contain negation, we will use the term "positive (propositional) program" (respectively, "positive (propositional) clause").

3.2 Stable Model Semantics

The stable model semantics [19] (see also [22]) is, along with program completion [14] and the well-founded semantics [18], one of the standard approaches to the semantics of general logic programs.

An interpretation under the stable model semantics is defined, as in the case of positive programs, as a mapping from B_P to the set $\mathbb{V}^2$ and identified with a subset of B_P. However, we need to extend the above definition, so that an interpretation can also assign meaning to negative literals.

Definition 10. Let I be a Herbrand interpretation of a given propositional program P. We extend I, so that for every negative literal $\sim$p, with $\mathsf{p} \in B_\mathsf{P}$:

$$\mathsf{I}(\sim\mathsf{p}) = \begin{cases} T, & \text{if } \mathsf{I}(\mathsf{p}) = F \\ F, & \text{if } \mathsf{I}(\mathsf{p}) = T \end{cases}$$

The concept of model is also defined in the same way as for positive programs.

The stable model semantics is defined through a derivative of a given logic program, called the *reduct* of the program:

Definition 11. Let P be a propositional program and let I be an interpretation of P. The *reduct* P_I of P with respect to I, is the set of clauses without negation that can be obtained from P by first dropping every clause of P that contains a negative literal $\sim p$ in its body such that $p \in I$, and then dropping all negative literals from the bodies of all the remaining clauses. The set I is called a *stable model* of P if I coincides with the least model of P_I.

It is shown in [19] that every stable model of a program P is indeed a model of P. However, it is not guaranteed that every program will have a unique stable model, as the following example demonstrates:

Example 7. The program

$$\begin{aligned} q &\leftarrow \\ p &\leftarrow \sim p \end{aligned}$$

does not have any stable models. On the other hand, the program:

$$\begin{aligned} q &\leftarrow \\ p &\leftarrow \sim q \end{aligned}$$

has exactly one stable model, namely $M = \{q\}$. Finally, the program:

$$\begin{aligned} p &\leftarrow \sim q \\ q &\leftarrow \sim p \end{aligned}$$

has two stable models, namely $M_1 = \{p\}$ and $M_2 = \{q\}$.

3.3 Well-founded Semantics

The well-founded semantics [18] is another major approach to the semantics of general programs. It differs from the stable model semantics in many ways; for example, under the well-founded semantics the meaning of a program is captured by a unique distinguished model and such a model exists for every logic program. Also, the semantics has a fixed-point characterisation which is an extension of the least fixed-point characterisation of the minimum model semantics for positive programs.

Perhaps the greatest difference between the stable model and the well-founded semantics, though, is that the latter is *three-valued*. This means that it uses an additional truth value, 0, which is stands for "unknown" and is assigned to propositional variables whose truth value cannot be decided under the semantics. We denote the new, three-valued truth domain $\{F, 0, T\}$ by $\mathbb{V}^3$.

We define two partial orders on $\mathbb{V}^3$. The first is denoted by $\leq$ and is the extension of the classical truth ordering. More specifically, $\leq$ is the partial order such that $F \leq F$, $0 \leq 0$, $T \leq T$ and $F \leq 0 \leq T$. The second is the *information* or *Fitting* ordering, denoted by $\preceq$ and defined by $F \preceq F$, $0 \preceq 0$, $T \preceq T$, $0 \preceq F$ and $0 \preceq T$.

Three-valued interpretations are defined analogously to two-valued interpretations:

Definition 12. A *three-valued Herbrand interpretation*, or *three-valued interpretation*, or *partial interpretation* I of a program P is a mapping from B_P to $\mathbb{V}^3$. Moreover, we define $I(\textit{true}) = T$ and $I(\textit{false}) = F$.

Definition 13. Let I be a three-valued interpretation of a given propositional program P. We extend I, so that for every negative literal $\sim$p, with p $\in B_{\mathsf{P}}$:

$$\mathsf{I}(\sim\mathsf{p}) = \begin{cases} T, & \text{if } \mathsf{I}(\mathsf{p}) = F \\ F, & \text{if } \mathsf{I}(\mathsf{p}) = T \\ 0, & \text{if } \mathsf{I}(\mathsf{p}) = 0 \end{cases}$$

Obviously, the set of propositional variables that are true under a three-valued interpretation does not suffice to fully represent the interpretation. Instead, we can denote a three-valued interpretation I as the pair $\langle T_{\mathsf{I}}, F_{\mathsf{I}} \rangle$, where T_{I} is the set of propositional variables that are assigned the value T and F_{I} is the set of propositional variables that are assigned the value F. The intersection of the sets T_{I} and F_{I} is of course empty, while their union is not necessarily the Herbrand base B_{P} of the program, hence the name "partial" interpretation. In contrast, a two-valued interpretation is also called a "total" interpretation and can be seen as a three-valued interpretation such that $T_{\mathsf{I}} \cup F_{\mathsf{I}} = B_{\mathsf{P}}$.

In this section, when we refer to an interpretation, we will always mean a partial interpretation, unless otherwise stated.

The concept of model is defined exactly as in the case of two-valued interpretations, based on the truth ordering on $\mathbb{V}^3$:

Definition 14. An interpretation M of a propositional program P is a model of P, if for every clause p $\leftarrow \mathsf{A}_1, \ldots, \mathsf{A}_m$ in P, it holds that $\mathsf{M}(\mathsf{p}) \geq min\{\mathsf{M}(\mathsf{A}_1), \ldots, \mathsf{M}(\mathsf{A}_m)\}$.

Our presentation of the construction of the well-founded semantics relies on the method of [28]. We first give the necessary definitions from [28].

Definition 15. Let P be a propositional program and let J be a partial interpretation of P. The operator $\Theta_{\mathsf{J}}(\cdot)$ on the set of partial interpretations of P is defined as follows: for every interpretation I and every propositional variable p of P,

$$\Theta_{\mathsf{J}}(\mathsf{I})(\mathsf{p}) = \begin{cases} T, & \text{there exists a clause p} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_n \text{ in P s.t. for all } i \leq n, \\ & \text{either } \mathsf{J}(\mathsf{L}_i) = T \text{ or } \mathsf{L}_i \text{ is a positive literal and } \mathsf{I}(\mathsf{L}_i) = T; \\ F, & \text{for all clauses p} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_n \text{ in P there exists an } i \leq n, \text{ s.t.} \\ & \text{either } \mathsf{J}(\mathsf{L}_i) = F \text{ or } \mathsf{L}_i \text{ is a positive literal and } \mathsf{I}(\mathsf{L}_i) = F; \\ 0, & \text{otherwise.} \end{cases}$$

Moreover we define the following sequence of interpretations:

$$\begin{array}{lcl} \Theta_{\mathsf{J}}^{\uparrow 0} & = & \langle T_0, F_0 \rangle = \langle \emptyset, B_{\mathsf{P}} \rangle \\ \Theta_{\mathsf{J}}^{\uparrow (n+1)} & = & \langle T_{n+1}, F_{n+1} \rangle = \Theta_{\mathsf{J}}(\Theta_{\mathsf{J}}^{\uparrow n}) \\ \Theta_{\mathsf{J}}^{\uparrow \omega} & = & \langle T_\omega, F_\omega \rangle = \langle \bigcup_{n<\omega} T_\beta, \bigcap_{n<\omega} F_\beta \rangle \end{array}$$

It is shown in [28] that, for any interpretation J, the operator Θ_{J} has a unique least fixed-point given by $\Theta_{\mathsf{J}}^{\uparrow \omega}$.

Definition 16. Let P be a propositional program. For every countable ordinal α, we define the interpretation M_α as follows:

$$\begin{array}{lcl} \mathsf{M}_0 & = & \langle T_0, F_0 \rangle = \langle \emptyset, \emptyset \rangle \\ \mathsf{M}_{\alpha+1} & = & \langle T_{\alpha+1}, F_{\alpha+1} \rangle = \Theta_{\mathsf{M}_\alpha}^{\uparrow \omega}, \text{ for a successor ordinal } \alpha + 1 \\ \mathsf{M}_\alpha & = & \langle T_\alpha, F_\alpha \rangle = \langle \bigcup_{\beta<\alpha} T_\beta, \bigcup_{\beta<\alpha} F_\beta \rangle, \text{ for a limit ordinal } \alpha \end{array}$$

Again from [28], there exists the least countable ordinal λ, such that $\mathsf{M}_\lambda = \Theta_{\mathsf{M}_\lambda}^{\uparrow\omega}$ and M_λ coincides with the well-founded model M_P of the propositional program P (originally defined in [18]).

Example 8. Cosnider again the following program from Example 7:

$$\begin{aligned} \mathtt{q} &\leftarrow \\ \mathtt{p} &\leftarrow \sim\mathtt{p} \end{aligned}$$

The well-founded model M_P of this program is the partial interpretation $\langle\{\mathtt{q}\},\{\}\rangle$, i.e. the interpretation such that $\mathsf{M}_\mathsf{P}(\mathtt{q}) = T$ and $\mathsf{M}_\mathsf{P}(\mathtt{p}) = 0$.

3.4 Locally Stratified Programs and the Perfect model semantics

There exists a broad class of general programs, called *locally stratified* programs, which have a clear, intuitive meaning. This is affirmed by the fact that, for locally stratified programs, the major semantic approaches, such as the well-founded and stable model semantics, always coincide. However, before the introduction of either the stable model or the well-founded semantics, the meaning of locally stratified programs was first captured by the perfect model semantics [1, 17].

The following definition of the local stratification of possibly infinite propositional programs is adapted to allow for the presence of the constants *true* and *false*.

Definition 17. A propositional program P is called *locally stratified* if and only if it is possible to decompose the Herbrand base B_P of P into disjoint sets (called *strata*) $S_1, S_2, \ldots, S_\alpha, \ldots, \alpha < \gamma$, where γ is a countable ordinal, such that for every clause $\mathsf{p} \leftarrow \mathsf{A}_1, \ldots, \mathsf{A}_m, \sim\mathsf{B}_1, \ldots, \sim\mathsf{B}_n$ in P, we have that for every $i \leq m$, ***stratum***$(\mathsf{A}_i) \leq$ ***stratum***(p) and for every $i \leq n$, ***stratum***$(\mathsf{B}_i) <$ ***stratum***(p), where ***stratum*** is a function such that ***stratum***$(\mathsf{C}) = \beta$, if the propositional variable $\mathsf{C} \in B_\mathsf{P}$ belongs to S_β and ***stratum***$(\mathsf{C}) = 0$, if $\mathsf{C} \notin B_\mathsf{P}$ and is a constant. Any decomposition of the described form is called a *local stratification* of P.

There exist a few alternative characterizations of the perfect model semantics (for example [1, 17, 29]), the most popular one probably being the preferred models characterisation from [29]. However, we choose to present here a fixed-point characterization given in [28], whose constructive nature is the basis for the proof of a main result in Section 5.3.1.

In [28], for any total interpretation J, the operator Ψ_J is defined and shown to have a unique least fixed-point given by $\Psi_\mathsf{J}^{\uparrow\omega}$ of the next definition. This is then used to give an iterated fixed-point characterization of the perfect model of a locally stratified program.

Definition 18. Let P be a propositional program and let J be a total interpretation of P. The operator $\Psi_\mathsf{J} : 2^{B_\mathsf{P}} \to 2^{B_\mathsf{P}}$ is defined as follows: for every $\mathsf{I} \subseteq B_\mathsf{P}$, $\Psi_\mathsf{J}(\mathsf{I}) = \{\mathsf{p} \in B_\mathsf{P} \mid$ there exists a clause $\mathsf{p} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_n$ in P such that, for all $i \leq n$, either $\mathsf{J}(\mathsf{L}_i) = T$ or $\mathsf{L}_i \in \mathsf{I}\}$. Moreover we define the following sequence:

$$\begin{aligned} \Psi_\mathsf{J}^{\uparrow 0} &= \emptyset \\ \Psi_\mathsf{J}^{\uparrow(n+1)} &= \Psi_\mathsf{J}(\Psi_\mathsf{J}^{\uparrow n}) \\ \Psi_\mathsf{J}^{\uparrow\omega} &= \bigcup_{n<\omega} \Psi_\mathsf{J}^{\uparrow n} \end{aligned}$$

Even though the interpretations defined above are total or two-valued interpretations and identified with subsets of the Herbrand base, we may use the partial interpretation notation and represent them as pairs. E.g., a set $\Psi_{\mathsf{J}}^{\uparrow n}$ of the above sequence may be represented as $\langle \Psi_{\mathsf{J}}^{\uparrow n}, B_{\mathsf{P}} - \Psi_{\mathsf{J}}^{\uparrow n} \rangle$.

Given a local stratification $S_1, S_2, \ldots, S_\alpha, \ldots$, $\alpha < \gamma$, of a propositional program P, we define the sets $\mathcal{B}_\alpha = \bigcup_{\beta<\alpha} S_\beta$ for every countable ordinal $\alpha \leq \gamma$. Clearly, $B_{\mathsf{P}} = \mathcal{B}_\gamma$. Then the perfect model of P can be constructed [28] as the last interpretation N_γ in an $\preceq$-increasing sequence of partial interpretations of P:

Definition 19. Let P be a propositional program and let $S_1, S_2, \ldots, S_\alpha, \ldots, \alpha < \gamma$, where γ is a countable ordinal, be a local stratification of P. For every countable ordinal $\alpha \leq \gamma$, we define the interpretation N_α as follows:

$$\begin{array}{lcl} N_0 & = & \langle T_0, F_0 \rangle = \langle \emptyset, \emptyset \rangle \\ N_{\alpha+1} & = & \langle T_{\alpha+1}, F_{\alpha+1} \rangle = \langle \Psi_{N_\alpha}^{\uparrow\omega}, \mathcal{B}_{\alpha+1} - \Psi_{N_\alpha}^{\uparrow\omega} \rangle, \text{ for a successor ordinal } \alpha+1 \\ N_\alpha & = & \langle T_\alpha, F_\alpha \rangle = \langle \bigcup_{\beta<\alpha} T_\beta, \bigcup_{\beta<\alpha} F_\beta \rangle, \text{ for a limit ordinal } \alpha \end{array}$$

Theorem 1 (Przymusinska, Przymusinski [28])**.** *Let* P *be a propositional program. The sequence* $N_0, N_1, \ldots, N_\alpha, \ldots, N_\gamma$ *is* $\preceq$*-increasing. Moreover,* N_γ *is the unique perfect model* N_{P} *of* P*.*

As implied earlier, both the stable model and the well-founded semantics agree with the perfect model semantics, when restricted to locally stratified programs.

Theorem 2 (Przymusinska, Przymusinski [27])**.** *Every locally stratified program* P *has a unique stable model which coincides with the perfect model of* P*.*

Theorem 3 (Van Gelder, Ross, Schlipf [18])**.** *The well-founded model of a locally stratified program* P *is two-valued and coincides with the perfect model of* P*.*

3.5 Infinite-valued Semantics

The infinite-valued semantics [30] is a far more recent approach to the semantics of general logic programs. The key idea of this approach is that, in order to give a logical semantics to negation-as-failure and to distinguish it from ordinary negation, one needs to extend the domain of truth values. For example, consider the program:

$$\begin{array}{l} \mathtt{p} \leftarrow \\ \mathtt{r} \leftarrow \sim\mathtt{p} \\ \mathtt{s} \leftarrow \sim\mathtt{q} \\ \mathtt{t} \leftarrow \sim\mathtt{t} \end{array}$$

According to negation-as-failure, both $\mathtt{p}$ and $\mathtt{s}$ receive the value T. However, $\mathtt{p}$ seems "truer" than $\mathtt{s}$ because there is a clause which says so, whereas $\mathtt{s}$ is true only because we are never obliged to make $\mathtt{q}$ true. In a sense, $\mathtt{s}$ is true only by default. For this reason, it was proposed in [30] to introduce a "default" truth value T_1 just below the "real" true T_0, and (by symmetry) a weaker false value F_1 just above ("not as false as") the real false F_0. Then, negation-as-failure is a combination of ordinary negation with a weakening. Thus $\sim F_0 = T_1$ and $\sim T_0 = F_1$. Since negations can be iterated, the new truth domain has a sequence $\ldots, T_3, T_2, T_1$ of weaker and weaker truth values below T_0 but above a neutral value 0; and a mirror image sequence $F_1, F_2, F_3, \ldots$ above F_0 and below 0. Since our

propositional programs are possibly countably infinite, we need a T_α and a F_α for every countable ordinal α. The intermediate truth value 0 is needed for certain atoms that have a "pathological" negative dependence on themselves (such as $\mathtt{t}$ in the above program). In conclusion, our truth domain $\mathbb{V}^\infty$ is shaped as follows:

$$F_0 < F_1 < \cdots < F_\omega < \cdots < F_\alpha < \cdots < 0 < \cdots < T_\alpha < \cdots < T_\omega < \cdots < T_1 < T_0$$

and the notion of "Herbrand interpretation of a program" can be generalized:

Definition 20. An (infinite-valued) *interpretation* I of a propositional program P is a mapping from B_P to the set $\mathbb{V}^\infty$. Moreover, we define $\mathsf{I}(\textit{true}) = T_0$ and $\mathsf{I}(\textit{false}) = F_0$.

For example, an infinite-valued interpretation for the program in the beginning of this section is $\mathsf{I} = \{(\mathtt{p}, T_3), (\mathtt{q}, F_0), (\mathtt{r}, 0), (\mathtt{s}, F_2), (\mathtt{t}, T_0)\}$. As we are going to see later in this section, the interpretation that captures the meaning of the above program is $\mathsf{J} = \{(\mathtt{p}, T_0), (\mathtt{q}, F_0), (\mathtt{r}, F_1), (\mathtt{s}, T_1), (\mathtt{t}, 0)\}$.

We will use $\emptyset$ to denote the infinite-valued interpretation that assigns the value F_0 to all propositional variables of a program. If $v \in \mathbb{V}^\infty$ is a truth value, we will use $\mathsf{I} \parallel v$ to denote the set of variables which are assigned the value v by I. In order to define the notion of "model", we need the following definitions:

Definition 21. Let I be an infinite-valued interpretation of a given propositional program P. For every negative literal $\sim$p appearing in P we extend I as follows:

$$\mathsf{I}(\sim\mathsf{p}) = \begin{cases} T_{\alpha+1}, & \text{if } \mathsf{I}(\mathsf{p}) = F_\alpha \\ F_{\alpha+1}, & \text{if } \mathsf{I}(\mathsf{p}) = T_\alpha \\ 0, & \text{if } \mathsf{I}(\mathsf{p}) = 0 \end{cases}$$

Moreover, for every conjunction of literals $\mathsf{L}_1, \ldots, \mathsf{L}_n$ appearing as the body of a clause in P, we extend I by $\mathsf{I}(\mathsf{L}_1, \ldots, \mathsf{L}_n) = min\{\mathsf{I}(\mathsf{L}_1), \ldots, \mathsf{I}(\mathsf{L}_n)\}$.

Definition 22. Let P be a propositional program and I an infinite-valued interpretation of P. Then, I *satisfies* a clause $\mathsf{p} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_n$ of P if $\mathsf{I}(\mathsf{p}) \geq \mathsf{I}(\mathsf{L}_1, \ldots, \mathsf{L}_n)$. Moreover, I is a *model* of P if I satisfies all clauses of P.

As it is demonstrated in [30], every program has a *minimum* infinite-valued model under an ordering relation $\sqsubseteq$, which compares interpretations in a stage-by-stage manner. To formally state this result, the following definitions are necessary:

Definition 23. The *order* of a truth value is defined as follows: $order(T_\alpha) = \alpha$, $order(F_\alpha) = \alpha$ and $order(0) = +\infty$.

Definition 24. Let I and J be infinite-valued interpretations of a given propositional program P and α be a countable ordinal. We write $\mathsf{I} =_\alpha \mathsf{J}$, if for all $\beta \leq \alpha$, $\mathsf{I} \parallel T_\beta = \mathsf{J} \parallel T_\beta$ and $\mathsf{I} \parallel F_\beta = \mathsf{J} \parallel F_\beta$. We write $\mathsf{I} \sqsubseteq_\alpha \mathsf{J}$, if for all $\beta < \alpha$, $\mathsf{I} =_\beta \mathsf{J}$ and, moreover, $\mathsf{I} \parallel T_\alpha \subseteq \mathsf{J} \parallel T_\alpha$ and $\mathsf{I} \parallel F_\alpha \supseteq \mathsf{J} \parallel F_\alpha$. We write $\mathsf{I} \sqsubset_\alpha \mathsf{J}$, if $\mathsf{I} \sqsubseteq_\alpha \mathsf{J}$ but $\mathsf{I} =_\alpha \mathsf{J}$ does not hold.

Definition 25. Let I and J be infinite-valued interpretations of a given propositional program P. We write $\mathsf{I} \sqsubset \mathsf{J}$, if there exists a countable ordinal α such that $\mathsf{I} \sqsubset_\alpha \mathsf{J}$. We write $\mathsf{I} \sqsubseteq \mathsf{J}$ if either $\mathsf{I} = \mathsf{J}$ or $\mathsf{I} \sqsubset \mathsf{J}$.

It is easy to see [30] that $\sqsubseteq$ is a partial order, $\sqsubseteq_\alpha$ is a preorder, and $=_\alpha$ is an equivalence relation. As in the case of positive programs, the minimum model of a program P coincides with the least fixed-point of an operator T_P. This operator is defined through the notion of the "least upper bound" of a set of truth values.

Definition 26. Let S be a set with a partial order $\leq$ and let $A \subseteq S$. We say that $u \in S$ is an *upper bound* of A, if for every $v \in A$ we have $v \leq u$. Moreover, u is called the *least upper bound* of A, if for every upper bound u' of A we have $u \leq u'$.

If the least upper bound of A exists then it is unique and we denote it by $lub(A)$. In [30] it is shown that every subset of $\mathbb{V}^\infty$ has a least upper bound and the operator T_P can then be defined as below:

Definition 27. Let P be a propositional program and let I be an interpretation of P. The *immediate consequence operator* T_P of P is defined as follows:

$$T_\mathsf{P}(\mathsf{I})(\mathsf{p}) = lub(\{\mathsf{I}(\mathsf{L}_1, \ldots, \mathsf{L}_n) \mid \mathsf{p} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_n \in \mathsf{P}\})$$

The least fixed-point M_P of T_P is constructed as follows. We start with $\emptyset$, namely the interpretation that assigns to every propositional variable of P the value F_0. We iterate T_P on $\emptyset$ until the set of variables having a F_0 value and the set of variables having a T_0 value, stabilize. Then we reset the values of all remaining variables to F_1. The procedure is repeated until the F_1 and T_1 values stabilize, and we reset the remaining variables to F_2, and so on. It is shown in [30] that there exists a countable ordinal δ for which this process will not produce any new variables having F_δ or T_δ values. At this point we reset all remaining variables to 0. The following definitions formalize this process.

Definition 28. Let P be a propositional program and let I be an infinite-valued interpretation of P. For each countable ordinal α, we define the infinite-valued interpretation $T^\omega_{\mathsf{P},\alpha}(\mathsf{I})$ as follows:

$$T^\omega_{\mathsf{P},\alpha}(\mathsf{I})(\mathsf{p}) = \begin{cases} \mathsf{I}(\mathsf{p}), & \text{if } order(\mathsf{I}(\mathsf{p})) < \alpha \\ T_\alpha, & \text{if } \mathsf{p} \in \bigcup_{n<\omega}(T^n_\mathsf{P}(\mathsf{I}) \parallel T_\alpha) \\ F_\alpha, & \text{if } \mathsf{p} \in \bigcap_{n<\omega}(T^n_\mathsf{P}(\mathsf{I}) \parallel F_\alpha) \\ F_{\alpha+1}, & \text{otherwise} \end{cases}$$

Definition 29. Let P be a propositional program. For each countable ordinal α, we define $\mathsf{M}_\alpha = T^\omega_{\mathsf{P},\alpha}(\mathsf{I}_\alpha)$ where $\mathsf{I}_0 = \emptyset$, $\mathsf{I}_\alpha = \mathsf{M}_{\alpha-1}$ if α is a successor ordinal, and

$$\mathsf{I}_\alpha(\mathsf{p}) \;=\; \begin{cases} \mathsf{M}_\beta(\mathsf{p}), & \text{if } order(\mathsf{M}_\beta(\mathsf{p})) = \beta \text{ for some } \beta < \alpha \\ F_\alpha, & \text{otherwise} \end{cases}$$

if α is a limit ordinal. The $\mathsf{M}_0, \mathsf{M}_1, \ldots, \mathsf{M}_\alpha, \ldots$ are called the *approximations* to the minimum model of P.

In [30] it is shown that the above sequence of approximations is well-defined. We will make use of the following lemma from [30]:

Lemma 1. *Let* P *be a propositional program and let* α *be a countable ordinal. For all* $n < \omega$, $T^n_\mathsf{P}(\mathsf{I}_\alpha) \sqsubseteq_\alpha \mathsf{M}_\alpha$.

The following lemma from [30] states that there exists a certain countable ordinal, after which new approximations do not introduce new truth values:

Lemma 2. *Let* P *be a propositional program. Then, there exists a countable ordinal* δ*, called the* depth *of* P*, such that:*

1. *for all countable ordinals* $\gamma \geq \delta$*,* $\mathsf{M}_\gamma \parallel T_\gamma = \emptyset$ *and* $\mathsf{M}_\gamma \parallel F_\gamma = \emptyset$*;*
2. *for all* $\beta < \delta$*,* $\mathsf{M}_\beta \parallel T_\beta \neq \emptyset$ *or* $\mathsf{M}_\beta \parallel F_\beta \neq \emptyset$*.*

Given a propositional program P that has depth δ, we define the following infinite-valued interpretation M_P:

$$\mathsf{M}_\mathsf{P}(\mathsf{p}) = \begin{cases} \mathsf{M}_\delta(\mathsf{p}), & \text{if } order(\mathsf{M}_\delta(\mathsf{p})) < \delta \\ 0, & \text{otherwise} \end{cases}$$

The following two theorems from [30], establish interesting properties of M_P:

Theorem 4. *The infinite-valued interpretation* M_P *is a model of* P*. Moreover, it is the least (with respect to* $\sqsubseteq$*) among all the infinite-valued models of* P*.*

Theorem 5. *The three-valued interpretation* N_P *obtained by collapsing all true values of* M_P *to* T *and all false values to* F*, coincides with the well-founded model of* P*.*

The next lemma states a fact already implied earlier, namely that new approximations do not affect the sets of variables stabilized by the preceding ones.

Lemma 3. *Let* P *be a propositional program and let* α *be a countable ordinal. For all countable ordinals* $\beta > \alpha$*,* $\mathsf{M}_\alpha =_\alpha \mathsf{M}_\beta$*. Moreover,* $\mathsf{M}_\alpha =_\alpha \mathsf{M}_\mathsf{P}$*.*

4. POSITIVE HIGHER-ORDER LOGIC PROGRAMS

In this chapter we define the syntax and semantics of positive higher-order logic programs. We present the two extensional approaches to the semantics of such programs, namely the domain-theoretic approach defined by W. W. Wadge in [31] and later extended and refined in [21, 9], on the one hand, and the more syntax-oriented approach taken by Bezem in [2, 3], on the other. Then we proceed to investigate the relationships between the two approaches and show that they coincide for the class of definitional programs, originally defined in [31]. Moreover, we show that this does not hold for the more general class of programs that we consider, i.e. programs with existentially quantified predicate variables in clause bodies. We identify the reasons behind this divergence and propose an additional assumption, under which the approaches continue to coincide even for programs with existential predicate variables.

4.1 Syntax

In this section we define the syntax of the higher-order language $\mathcal{H}$. This language is a strict superset of the language considered by Wadge in [31] and a strict subset of the language of Bezem [3].

$\mathcal{H}$ is based on a simple type system with two base types: o, the boolean domain, and ι, the domain of data objects. The composite types are partitioned into three classes: functional (assigned to function symbols), predicate (assigned to predicate symbols) and argument (assigned to parameters of predicates).

Definition 30. A type can either be *functional*, *predicate*, or *argument*, denoted by σ, π and ρ respectively and defined as:

$$\sigma := \iota \mid (\iota \to \sigma)$$
$$\pi := o \mid (\rho \to \pi)$$
$$\rho := \iota \mid \pi$$

We will use τ to denote an arbitrary type (either functional, predicate, or argument). As usual, the binary operator $\to$ is right-associative. A functional type that is different than ι will often be written in the form $\iota^n \to \iota$, $n \geq 1$. Moreover, it can be easily seen that every predicate type π can be written in the form $\rho_1 \to \cdots \to \rho_n \to o$, $n \geq 0$ (for $n = 0$ we assume that $\pi = o$). We proceed by defining the syntax of $\mathcal{H}$:

Definition 31. The *alphabet* of $\mathcal{H}$ consists of the following:

- *predicate variables* of every predicate type π (denoted by capital letters such as $\mathsf{Q}, \mathsf{R}, \mathsf{S}, \ldots$);
- *individual variables* of type ι (denoted by capital letters such as $\mathsf{X}, \mathsf{Y}, \mathsf{Z}, \ldots$);
- *predicate constants* of every predicate type π (denoted by lowercase letters such as $\mathsf{p}, \mathsf{q}, \mathsf{r}, \ldots$);
- *individual constants* of type ι (denoted by lowercase letters such as $\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots$);
- *function symbols* of every functional type $\sigma \neq \iota$ (denoted by lowercase letters such as $\mathsf{f}, \mathsf{g}, \mathsf{h}, \ldots$);

- the *inverse implication* constant $\leftarrow$; the comma; the left and right parentheses; and the *equality* constant $\approx$ for comparing terms of type ι.

Arbitrary variables will be usually denoted by V and its subscripted versions.

Definition 32. The set of *terms* of $\mathcal{H}$ is defined as follows:

- every predicate variable (respectively, predicate constant) of type π is a term of type π;
- every individual variable (respectively, individual constant) of type ι is a term of type ι;
- if f is an n-ary function symbol and $\mathsf{E}_1, \ldots, \mathsf{E}_n$ are terms of type ι then $(\mathsf{f}\ \mathsf{E}_1 \cdots \mathsf{E}_n)$ is a term of type ι;
- if E_1 is a term of type $\rho \to \pi$ and E_2 a term of type ρ then $(\mathsf{E}_1\ \mathsf{E}_2)$ is a term of type π.

Terms of type o will also be referred to as *atoms*.

Definition 33. The set of *expressions* of $\mathcal{H}$ is defined as follows:

- A term of type ρ is an expression of type ρ;
- if E_1 and E_2 are terms of type ι, then $(\mathsf{E}_1 \approx \mathsf{E}_2)$ is an expression of type o.

We will omit parentheses when no confusion arises. To denote that an expression E has type ρ we will often write $\mathsf{E} : \rho$. We will write $vars(\mathsf{E})$ to denote the set of all the variables in E. Expressions (respectively, terms) that have no variables will be referred to as *ground expressions* (respectively, *ground terms*).

Next, we define the set of $\mathcal{H}$ programs and its subset of definitional programs.

Definition 34. A *clause* of $\mathcal{H}$ is a formula $\mathsf{p}\ \mathsf{V}_1 \cdots \mathsf{V}_n \leftarrow \mathsf{E}_1, \ldots, \mathsf{E}_m$, where p is a predicate constant of type $\rho_1 \to \cdots \to \rho_n \to o$, $\mathsf{V}_1, \ldots, \mathsf{V}_n$ are distinct variables of types $\rho_1, \ldots, \rho_n$ respectively and $\mathsf{E}_1, \ldots, \mathsf{E}_m$ are expressions of type o. The term $\mathsf{p}\ \mathsf{V}_1 \cdots \mathsf{V}_n$ is called the *head* of the clause, the variables $\mathsf{V}_1, \ldots, \mathsf{V}_n$ are the *formal parameters* of the clause and the conjunction $\mathsf{E}_1, \ldots, \mathsf{E}_m$ is its *body*.

A *definitional clause* is a clause that additionally satisfies the following restriction: the only predicate variables that can appear in the body of the clause are its formal parameters.

A *program* P of $\mathcal{H}$ is a finite set of clauses. A *definitional program* is a finite set of definitional clauses.

Notice that for uniformity reasons, the above definition requires that *all* formal parameters are distinct, even the type ι ones. This is not a restriction, because two occurrences of the same individual variable in the head of a clause can be replaced by distinct variables, which are then explicitly equated in the body of the clause using the constant $\approx$ (see Example 9 that follows).

We will often refrain from referring to the language $\mathcal{H}$ and we will usually talk with respect to a given program P. For example, when we are considering a given program P and

discuss an expression E, we will always mean an expression that can be created with syntactic elements that appear specifically in P (and not more generally with elements that appear in $\mathcal{H}$).

The syntax of programs given in Definition 34 differs slightly from the Prolog-like syntax that we have used in Chapter 1. However, one can easily verify that we can transform every program from the former syntax to the latter.

Example 9. Consider the following program in Prolog-like syntax:

```
p(a).
q(X,X).
r(P,Q,f(X)):-P(X),Q(Y).
```

In our more formal notation, this can be written as:

```
p X ← (X ≈ a)
q X Y ← (X ≈ Y)
r P Q Z ← (Z ≈ f(X)),(P X),(Q Y)
```

Notice that the formal parameters of every clause are now distinct. □

The ground instantiation of a program is described by the following definitions:

Definition 35. A *substitution* θ is a finite set of the form $\{\mathsf{V}_1/\mathsf{E}_1, \ldots, \mathsf{V}_n/\mathsf{E}_n\}$ where the V_i's are different variables and each E_i is a term having the same type as V_i. We write $dom(\theta)$ to denote the domain $\{\mathsf{V}_1, \ldots, \mathsf{V}_n\}$ of θ. If all the terms $\mathsf{E}_1, \ldots, \mathsf{E}_n$ are ground, θ is called a *ground substitution*.

Definition 36. Let θ be a substitution and E be an expression. Then, Eθ is an expression obtained from E as follows:

- $\mathsf{E}\theta = \mathsf{E}$ if E is a predicate constant or individual constant;
- $\mathsf{V}\theta = \theta(\mathsf{V})$ if $\mathsf{V} \in dom(\theta)$; otherwise, $\mathsf{V}\theta = \mathsf{V}$;
- $(\mathsf{f}\ \mathsf{E}_1 \cdots \mathsf{E}_n)\theta = (\mathsf{f}\ \mathsf{E}_1\theta \cdots \mathsf{E}_n\theta)$;
- $(\mathsf{E}_1\ \mathsf{E}_2)\theta = (\mathsf{E}_1\theta\ \mathsf{E}_2\theta)$;
- $(\mathsf{E}_1 \approx \mathsf{E}_2)\theta = (\mathsf{E}_1\theta \approx \mathsf{E}_2\theta)$.

If θ is a ground substitution such that $vars(\mathsf{E}) \subseteq dom(\theta)$, then the ground expression Eθ is called a *ground instance* of E.

In the case of first-order logic programs, the Herbrand Universe comprises ground terms of type ι. In the higher-order setting a Herbrand Universe is needed for each argument type ρ.

Definition 37. For a program P, we define the *Herbrand universe* for every argument type ρ, denoted by $U_{\mathsf{P},\rho}$, to be the set of all ground terms of type ρ that can be formed out of the individual constants, function symbols and predicate constants in the program. Moreover, we define $U^+_{\mathsf{P},o}$ to be the set of all ground expressions of type o, that can be formed out of the above symbols, i.e. the set $U^+_{\mathsf{P},o} = U_{\mathsf{P},o} \cup \{(\mathsf{E}_1 \approx \mathsf{E}_2) \mid \mathsf{E}_1, \mathsf{E}_2 \in U_{\mathsf{P},\iota}\}$.

Definition 38. Let P be a program. A *ground instance of a clause* $\mathsf{p}\ \mathsf{V}_1 \cdots \mathsf{V}_n \leftarrow \mathsf{E}_1, \ldots, \mathsf{E}_m$ of P is a formula $(\mathsf{p}\ \mathsf{V}_1 \cdots \mathsf{V}_n)\theta \leftarrow \mathsf{E}_1\theta, \ldots, \mathsf{E}_m\theta$, where θ is a ground substitution whose domain is the set of all variables that appear in the clause, such that for every $\mathsf{V} \in dom(\theta)$ with $\mathsf{V} : \rho$, it is $\theta(\mathsf{V}) \in U_{\mathsf{P},\rho}$. The *ground instantiation of a program* P, denoted by $\mathrm{Gr}(\mathsf{P})$, is the (possibly infinite) set that contains all the ground instances of the clauses of P.

4.2 Domain Theoretic Semantics

In this section we introduce the main concepts behind Wadge's semantics of [31] as they were extended and refined in [21]. The key idea behind this semantics is (intuitively) to assign to program predicates monotonic relations.

Note that the original semantics of [31] was defined on the class of definitional programs. However, it straightforwardly extends to apply to our $\mathcal{H}$ programs, i.e., all the results mentioned in this section are easily shown to also hold for programs with existential predicate variables. Actually, the language HOPES [9] that has been built based on Wadge's semantics, makes heavy use of programs that allow existential predicate variables in bodies of clauses or in queries.

Given a program P of $\mathcal{H}$, we define the semantics of types with respect to the Herbrand universe $U_{\mathsf{P},\iota}$ of P. More specifically, we define simultaneously and recursively two things: the semantics $[\![\tau]\!]^W$[2] of a type τ and a corresponding partial order $\leq_\tau^W$ on the elements of $[\![\tau]\!]^W$. Given posets A and B, we write $[A \xrightarrow{m} B]$ to denote the set of all monotonic functions from A to B.

Definition 39. Let P be a program. Then,

- $[\![\iota]\!]^W = U_{\mathsf{P},\iota}$ and $\leq_\iota^W$ is the trivial partial order that relates every element to itself;
- $[\![\iota^n \to \iota]\!]^W = U_{\mathsf{P},\iota}^n \to U_{\mathsf{P},\iota}$. A partial order for this case is not needed;
- $[\![o]\!]^W = \mathbb{V}^2$ and $\leq_o^W$ is the partial order $\leq$ on truth values;
- $[\![\rho \to \pi]\!]^W = [[\![\rho]\!]^W \xrightarrow{m} [\![\pi]\!]^W]$ and $\leq_{\rho\to\pi}^W$ is the partial order defined as follows: for all $f, g \in [\![\rho \to \pi]\!]^W$, $f \leq_{\rho\to\pi}^W g$ iff $f(d) \leq_\pi^W g(d)$ for all $d \in [\![\rho]\!]^W$.

We now proceed to define Herbrand interpretations and states.

Definition 40. Let P be a program. A *Herbrand interpretation I of* P consists of the following assignments:

1. to each individual constant c that appears in P, of the element $I(\mathsf{c}) = \mathsf{c}$;
2. to each predicate constant $\mathsf{p} : \pi$ that appears in P, of an element $I(\mathsf{p}) \in [\![\pi]\!]^W$;
3. to each n-ary function symbol f that appears in P, of the element $I(\mathsf{f}) \in [\![\iota^n \to \iota]\!]^W$ such that for all $\mathsf{t}_1, \ldots, \mathsf{t}_n \in U_{\mathsf{P},\iota}$, $I(\mathsf{f})\ \mathsf{t}_1 \cdots \mathsf{t}_n = \mathsf{f}\ \mathsf{t}_1 \cdots \mathsf{t}_n$.

[2]The superscript W, which annotates various symbols used in this section, stands for "Wadge". It is used to distinguish these symbols from the respective symbols (introduced in the next section) for analogous concepts of Bezem's semantics.

Definition 41. A *Herbrand state* s of a program P is a function that assigns to each argument variable V of type ρ, an element $s(\mathsf{V}) \in [\![\rho]\!]^W$.

A (Herbrand) interpretation under Wadge's semantics may be referred to as a "domain-theoretic (Herbrand) interpretation", when we need to distinguish it from other types of higher-order interpretations that we will present in the following sections. On the other hand, it may be referred to simply as a "(Herbrand) interpretation" when no confusion arises, e.g. in the remainder of the present section. Similarly, a state of the above form may be referred to as a "domain-theoretic (Herbrand) state" in the former situation, or simply a "(Herbrand) state" in the latter.

In the following, $s[\mathsf{V}_1/d_1, \ldots, \mathsf{V}_n/d_n]$ is used to denote a state that is identical to s the only difference being that the new state assigns to each V_i the corresponding value d_i.

Definition 42. Let P be a program, I be a Herbrand interpretation of P and s be a Herbrand state. Then, the semantics of expressions is defined as follows:

1. $[\![\mathsf{V}]\!]^W_s(I) = s(\mathsf{V})$ if V is a variable;
2. $[\![\mathsf{c}]\!]^W_s(I) = I(\mathsf{c})$ if c is an individual constant;
3. $[\![\mathsf{p}]\!]^W_s(I) = I(\mathsf{p})$ if p is a predicate constant;
4. $[\![(\mathsf{f}\ \mathsf{E}_1 \cdots \mathsf{E}_n)]\!]^W_s(I) = I(\mathsf{f})\ [\![\mathsf{E}_1]\!]^W_s(I) \cdots [\![\mathsf{E}_n]\!]^W_s(I)$;
5. $[\![(\mathsf{E}_1\ \mathsf{E}_2)]\!]^W_s(I) = [\![\mathsf{E}_1]\!]^W_s(I)\ [\![\mathsf{E}_2]\!]^W_s(I)$;
6. $[\![(\mathsf{E}_1 \approx \mathsf{E}_2)]\!]^W_s(I) = T$ if $[\![\mathsf{E}_1]\!]^W_s(I) = [\![\mathsf{E}_2]\!]^W_s(I)$ and F otherwise.

For ground expressions E we will often write $[\![\mathsf{E}]\!]^W(I)$ instead of $[\![\mathsf{E}]\!]^W_s(I)$ since in this case the meaning of E is independent of s.

It is straightforward to confirm that the above definition assigns to every expression an element of the corresponding semantic domain, as stated in the following lemma:

Lemma 4. *Let* P *be a program and let* $\mathsf{E} : \rho$ *be an expression. Also, let I be a Herbrand interpretation and s be a Herbrand state. Then* $[\![\mathsf{E}]\!]^W_s(I) \in [\![\rho]\!]^W$.

Definition 43. Let P be a program and M be a Herbrand interpretation of P. Then, M is a *Herbrand model* of P iff for every clause $\mathsf{p}\ \mathsf{V}_1 \cdots \mathsf{V}_n \leftarrow \mathsf{E}_1, \ldots, \mathsf{E}_m$ in P and for every Herbrand state s, it holds $[\![\mathsf{p}\ \mathsf{V}_1 \cdots \mathsf{V}_n]\!]^W_s(M) \geq min\{[\![\mathsf{E}_1]\!]^W_s(M), \ldots, [\![\mathsf{E}_m]\!]^W_s(M)\}$.

In the following we denote the set of Herbrand interpretations of a program P with Int_{P}. We define a partial order on Int_{P} as follows: for all $I, J \in Int_{\mathsf{P}}$, $I \leq^W_{Int_{\mathsf{P}}} J$ iff for every predicate constant $\mathsf{p} : \pi$ that appears in P, $I(\mathsf{p}) \leq^W_\pi J(\mathsf{p})$. Similarly, we denote the set of Herbrand states with S_{P} and we define a partial order as follows: for all $s_1, s_2 \in S_{\mathsf{P}}$, $s_1 \leq^W_{S_{\mathsf{P}}} s_2$ iff for all variables $\mathsf{V} : \rho$, $s_1(\mathsf{V}) \leq^W_\rho s_2(\mathsf{V})$. The following lemmata are straightforward to establish:

Lemma 5. *Let* P *be a program. Then,* $(Int_{\mathsf{P}}, \leq^W_{Int_{\mathsf{P}}})$ *is a complete lattice.*

Lemma 6. *Let* P *be a program and let* $\mathsf{E} : \rho$ *be an expression. Let* I, J *be Herbrand interpretations and* s, s' *be Herbrand states. Then,*

1. *If* $I \leq^W_{Int_\mathsf{P}} J$ *then* $[\![\mathsf{E}]\!]^W_s(I) \leq^W_\rho [\![\mathsf{E}]\!]^W_s(J)$.
2. *If* $s \leq^W_{S_\mathsf{P}} s'$ *then* $[\![\mathsf{E}]\!]^W_s(I) \leq^W_\rho [\![\mathsf{E}]\!]^W_{s'}(I)$.

We can now define the *immediate consequence operator* for $\mathcal{H}$ programs, which generalizes the corresponding operator for classical (first-order) programs [23].

Definition 44. Let P be a program. The mapping $T_\mathsf{P} : Int_\mathsf{P} \to Int_\mathsf{P}$ is called the *immediate consequence operator for* P and is defined for every predicate constant $\mathsf{p} : \rho_1 \to \cdots \to \rho_n \to o$ and $d_i \in [\![\rho_i]\!]^W$ as

$$T_\mathsf{P}(I)(\mathsf{p})\ d_1 \cdots d_n = \begin{cases} T, & \text{if there exists a clause } \mathsf{p}\ \mathsf{V}_1 \cdots \mathsf{V}_n \leftarrow \mathsf{E}_1, \ldots, \mathsf{E}_m \text{ and} \\ & \text{a Herbrand state } s \text{ such that } [\![\mathsf{E}_i]\!]^W_{s[\mathsf{V}_1/d_1,\ldots,\mathsf{V}_n/d_n]}(I) = T \\ & \text{for all } i \in \{1, \ldots, m\} \\ F, & \text{otherwise.} \end{cases}$$

It is not hard to see that T_P is a monotonic function, and this leads to the following theorem [31, 21]:

Theorem 6. *Let* P *be a program. Then* $M_\mathsf{P} = lfp(T_\mathsf{P})$ *is the minimum, with respect to* $\leq^W_{Int_\mathsf{P}}$*, Herbrand model of* P*.*

4.3 Bezem's Semantics

M. Bezem [2, 3] developed a semantics for higher-order logic programs which generalizes the familiar Herbrand-model semantics of classical (first-order) logic programs. Note that the class of programs considered in [2, 3] (named *hoapata* programs) is a strict superset of the class defined in Section 4.1. In particular, the head of a clause in a hoapata program, may be an atom whose left-most predicate is a variable. However, we will restrict attention to $\mathcal{H}$ programs.

We begin by specifying the semantic domains in which the expressions of each type τ are assigned their meanings. The following definition is a slightly modified version of the corresponding definition of Bezem [2, 3] and it implies that the expressions of predicate types should be understood as representing functions. We use $[S_1 \to S_2]$ to denote the set of (possibly partial) functions from a set S_1 to a set S_2. The possibility to have a partial function arises due to a technicality which is explained in the remark just above Definition 46.

Definition 45. A *functional type structure* $\mathbb{S}$ for $\mathcal{H}$ consists of two non-empty sets D and A together with an assignment $[\![\tau]\!]^B$[3] to each type τ of $\mathcal{H}$, so that the following are satisfied:

- $[\![\iota]\!]^B = D$;
- $[\![\iota^n \to \iota]\!]^B = D^n \to D$;
- $[\![o]\!]^B = A$;

[3]The superscript B stands for "Bezem".

- $[\![\rho \to \pi]\!]^B \subseteq [[\![\rho]\!]^B \to [\![\pi]\!]^B]$.

Given a functional type structure $\mathbb{S}$, any function *val* : $[\![o]\!]^B \to \mathbb{V}^2$ will be called a *(two-valued) valuation function* for $\mathbb{S}$.

As in the previous section, we apply the usual practice in the study of the semantics of logic programming languages and restrict attention to interpretations that have the Herbrand universe of the program as their underlying universe. Following Bezem [2, 3], we take D and A in Definition 45 to be equal to $U_{\mathsf{P},\iota}$ and $U^+_{\mathsf{P},o}$ respectively. Then, for each predicate type $\rho \to \pi$, each element of $U_{\mathsf{P},\rho\to\pi}$ can be perceived as a function mapping elements of $[\![\rho]\!]^B$ to elements of $[\![\pi]\!]^B$, through syntactic application mapping. That is, $\mathsf{E} \in U_{\mathsf{P},\rho\to\pi}$ can be viewed as the function mapping each term $\mathsf{E}' \in U_{\mathsf{P},\rho}$ to the term $(\mathsf{E}\,\mathsf{E}') \in U_{\mathsf{P},\pi}$. Similarly, every n-ary function symbol f appearing in P can be viewed as the function mapping each element $(\mathsf{E}_1, \ldots, \mathsf{E}_n) \in U^n_{\mathsf{P},\iota}$ to the term $(\mathsf{f}\,\mathsf{E}_1 \cdots \mathsf{E}_n) \in U_{\mathsf{P},\iota}$.

Remark: There is a small technicality here which we need to clarify. In the case where $\rho = o$, $\mathsf{E} \in U_{\mathsf{P},o\to\pi}$ is a partial function because it maps elements of $U_{\mathsf{P},o}$ (and not of $U^+_{\mathsf{P},o}$) to elements of $U_{\mathsf{P},\pi}$; this is due to the fact that our syntax does not allow an expression of type $o \to \pi$ to take as argument an expression of the form $(\mathsf{E}_1 \approx \mathsf{E}_2)$ (nor negated atoms – see Chapter 5). In all other cases (i.e., when $\rho \neq o$), E represents a total function.

We can now define the notion of (Herbrand) interpretation under Bezem's semantics. We may also refer to such an interpretation as a "Bezem-type (Herbrand) interpretation" or simply as a "(Herbrand) interpretation", when the type of interpretation is obvious from context.

Definition 46. A *Herbrand interpretation* $\mathcal{I}$ of a program P consists of:

1. the functional type structure $\mathbb{S}_\mathsf{P}$, such that $D = U_{\mathsf{P},\iota}$, $A = U^+_{\mathsf{P},o}$ and $[\![\rho \to \pi]\!]^B = U_{\mathsf{P},\rho\to\pi}$ for every predicate type $\rho \to \pi$, called the Herbrand type structure of P;
2. the assignment to each individual constant c in P, of the element $\mathcal{I}(\mathsf{c}) = \mathsf{c}$; to each predicate constant p in P, of the element $\mathcal{I}(\mathsf{p}) = \mathsf{p}$; to each function symbol f in P, of the element $\mathcal{I}(\mathsf{f}) = \mathsf{f}$;
3. a valuation function $\mathit{val}_\mathcal{I}(\cdot)$ for $\mathbb{S}_\mathsf{P}$, assigning to each element of $U^+_{\mathsf{P},o}$ an element in $\mathbb{V}^2$, while satisfying:

$$\mathit{val}_\mathcal{I}((\mathsf{E}_1 \approx \mathsf{E}_2)) = \begin{cases} F, & \text{if } \mathsf{E}_1 \neq \mathsf{E}_2 \\ T, & \text{if } \mathsf{E}_1 = \mathsf{E}_2 \end{cases}.$$

   for all $\mathsf{E}_1, \mathsf{E}_2 \in U_{\mathsf{P},\iota}$.

We call $\mathit{val}_\mathcal{I}(\cdot)$ the *valuation function of* $\mathcal{I}$ and omit the reference to $\mathbb{S}_\mathsf{P}$, since the latter is common to all Herbrand interpretations of a program. In fact, individual Herbrand interpretations are only set apart by their valuation functions. As in Section 4.2, a (Herbrand) state is again defined as a function that assigns to each argument variable of the program, an element in the appropriate semantic domain.

Definition 47. A *Herbrand state* s of a program P is a function that assigns to each variable V of type ρ an element of $U_{\mathsf{P},\rho}$.

Given a Herbrand interpretation $\mathcal{I}$ and state $\mathcal{s}$, we can define the semantics of expressions with respect to $\mathcal{I}$ and $\mathcal{s}$.

Definition 48. Let P be a program. Also, let $\mathcal{I}$ be a Herbrand interpretation and $\mathcal{s}$ a Herbrand state of P. Then the semantics of expressions with respect to $\mathcal{I}$ and $\mathcal{s}$ is defined as follows:

- $[\![\mathsf{c}]\!]^B_{\mathcal{s}}(\mathcal{I}) = \mathcal{I}(\mathsf{c}) = \mathsf{c}$, for every individual constant c;
- $[\![\mathsf{p}]\!]^B_{\mathcal{s}}(\mathcal{I}) = \mathcal{I}(\mathsf{p}) = \mathsf{p}$, for every predicate constant p;
- $[\![\mathsf{V}]\!]^B_{\mathcal{s}}(\mathcal{I}) = \mathcal{s}(\mathsf{V})$, for every variable V;
- $[\![(\mathsf{f}\,\mathsf{E}_1 \cdots \mathsf{E}_n)]\!]^B_{\mathcal{s}}(\mathcal{I}) = (\mathcal{I}(\mathsf{f})\,[\![\mathsf{E}_1]\!]^B_{\mathcal{s}}(\mathcal{I}) \cdots [\![\mathsf{E}_n]\!]^B_{\mathcal{s}}(\mathcal{I})) = (\mathsf{f}\,[\![\mathsf{E}_1]\!]^B_{\mathcal{s}}(\mathcal{I}) \cdots [\![\mathsf{E}_n]\!]^B_{\mathcal{s}}(\mathcal{I}))$, for every n-ary function symbol f;
- $[\![(\mathsf{E}_1\,\mathsf{E}_2)]\!]^B_{\mathcal{s}}(\mathcal{I}) = ([\![\mathsf{E}_1]\!]^B_{\mathcal{s}}(\mathcal{I})\,[\![\mathsf{E}_2]\!]^B_{\mathcal{s}}(\mathcal{I}))$;
- $[\![(\mathsf{E}_1 \approx \mathsf{E}_2)]\!]^B_{\mathcal{s}}(\mathcal{I}) = ([\![\mathsf{E}_1]\!]^B_{\mathcal{s}}(\mathcal{I}) \approx [\![\mathsf{E}_2]\!]^B_{\mathcal{s}}(\mathcal{I}))$.

It is easy to see that the semantic function $[\![\cdot]\!]^B$ is well defined, in the sense that, for every Herbrand state $\mathcal{s}$ and every expression E of every argument type ρ, we have $[\![\mathsf{E}]\!]^B_{\mathcal{s}}(\mathcal{I}) \in [\![\rho]\!]^B$. Note that this makes $[\![\mathsf{E}]\!]^B_{\mathcal{s}}(\mathcal{I})$ a ground expression of the language. Also, note that if E is a ground expression then $[\![\mathsf{E}]\!]^B_{\mathcal{s}}(\mathcal{I}) = \mathsf{E}$; therefore, if E is moreover of type o, we can write $\mathit{val}_{\mathcal{I}}(\mathsf{E})$ instead of $\mathit{val}_{\mathcal{I}}([\![\mathsf{E}]\!]^B_{\mathcal{s}}(\mathcal{I}))$. Stretching this abuse of notation a little further, we can extend a valuation function to assign truth values to ground conjunctions of literals; this allows us to define the concept of Herbrand models for our higher-order programs in the same way as in classical logic programming.

Definition 49. Let P be a program and $\mathcal{I}$ be a Herbrand interpretation of P. We define $\mathit{val}_{\mathcal{I}}(\mathsf{E}_1, \ldots, \mathsf{E}_n) = min\{\mathit{val}_{\mathcal{I}}(\mathsf{E}_1), \ldots, \mathit{val}_{\mathcal{I}}(\mathsf{E}_n)\}$ for all $\mathsf{E}_1, \ldots, \mathsf{E}_n \in U^+_{\mathsf{P},o}$. Moreover, we say $\mathcal{I}$ is a *model* of P if $\mathit{val}_{\mathcal{I}}([\![\mathsf{A}]\!]^B_{\mathcal{s}}(\mathcal{I})) \geq \mathit{val}_{\mathcal{I}}([\![\mathsf{E}_1]\!]^B_{\mathcal{s}}(\mathcal{I}), \ldots, [\![\mathsf{E}_m]\!]^B_{\mathcal{s}}(\mathcal{I}))$ holds for every clause $\mathsf{A} \leftarrow \mathsf{E}_1, \ldots, \mathsf{E}_m$ and every Herbrand state $\mathcal{s}$ of P.

Bezem's semantics is based on the observation that, given a positive higher-order program P, we can use the minimum model of its ground instantiation as a (two-valued) valuation function defining a Herbrand interpretation $\mathcal{M}_{\mathsf{P}}$ for the initial program itself.

Remark: One small concern when using the minimum model of the ground instantiation as a valuation function, is the way in which expressions of the form $(\mathsf{E}_1 \approx \mathsf{E}_2)$ should be handled. Recall that Definition 46 requires that such atoms are given fixed truth values by all Herbrand interpretations, so they cannot be treated as propositional variables. Instead, it makes more sense that they are treated as logical constants, by the following convention: for all $\mathsf{E}_1, \mathsf{E}_2 \in U_{\mathsf{P},\iota}$, $(\mathsf{E}_1 \approx \mathsf{E}_2)$ is equivalent to the constant *true* if $\mathsf{E}_1 = \mathsf{E}_2$, otherwise it is equivalent to the constant *false*. This way, every interpretation of the ground program will assign the truth value T in the first case and the value F in the second, conforming to the restrictions imposed by Definition 46. This technicality is the reason behind allowing the logical constants to appear in clause bodies in propositional programs.

Definition 50. Let P be a program and let $\mathsf{Gr}(\mathsf{P})$ be the ground instantiation of P. Also, let $\mathsf{M}_{\mathsf{Gr}(\mathsf{P})}$ be the minimum model of $\mathsf{Gr}(\mathsf{P})$. Then $\mathcal{M}_{\mathsf{P}}$ is the Herbrand interpretation of P such that for every $\mathsf{A} \in U_{\mathsf{P},o}$, $\mathit{val}_{\mathcal{M}_{\mathsf{P}}}(\mathsf{A}) = \mathsf{M}_{\mathsf{Gr}(\mathsf{P})}(\mathsf{A})$.

A main contribution of [3] is demonstrating that, for $\mathcal{H}$ (or more generally, hoapata) programs, $\mathcal{M}_\mathsf{P}$ is a model of the higher-order program. Moreover, Bezem showed that the above model is also monotonic [4], extensional [2, 3] and the minimum model of P [2, 3]. These three properties are discussed in Section 4.3.1 that follows.

4.3.1 Properties of $\mathcal{M}_\mathsf{P}$

The notion of minimum model is defined (in the usual way) with respect to $\leq^B_{\mathcal{I}_\mathsf{P}}$, i.e. an ordering on the set of interpretations $\mathcal{I}_\mathsf{P}$ of a given program P based on the standard truth ordering $\leq$.

Definition 51. If $\mathcal{I}$ and $\mathcal{J}$ are two Herbrand interpretations of a higher-order program P, we say $\mathcal{I} \leq^B_{\mathcal{I}_\mathsf{P}} \mathcal{J}$ if, for all atoms A in $U_{\mathsf{P},o}$ we have $\mathit{val}_\mathcal{I}(\mathsf{A}) \leq \mathit{val}_\mathcal{J}(\mathsf{A})$.

The minimum model property of $\mathcal{M}_\mathsf{P}$, stated in the following theorem, is a special case of a more general result that is given in [3] (Lemma 2, Case 3, page 209).

Theorem 7 (Minimum model property)**.** $\mathcal{M}_\mathsf{P}$ *is the minimum, with respect to* $\leq^B_{\mathcal{I}_\mathsf{P}}$*, Herbrand model of* P*.*

The extensionality property is formally defined in [2, 3] through relations $\cong_{val,\tau}$ over the set of expressions of a given type τ and under a given valuation function val. These relations intuitively express extensional equality of type τ, in the sense discussed in Section 1.5. For our purposes, only extensional equality of argument types will be needed, for which we give a slightly modified definition below:

Definition 52. Let $\mathbb{S}$ be a functional type structure and val be a valuation function for $\mathbb{S}$. For every argument type ρ we define the relations $\cong_{val,\rho}$ on $[\![\rho]\!]^B$ as follows: Let $d, d' \in [\![\rho]\!]^B$; then $d \cong_{val,\rho} d'$ if and only if

1. $\rho = \iota$ and $d = d'$, or
2. $\rho = o$ and $val(d) = val(d')$, or
3. $\rho = \rho' \to \pi$ and $d\,e \cong_{val,\pi} d'\,e'$ for all $e, e' \in [\![\rho']\!]^B$, such that $e \cong_{val,\rho'} e'$ and $d\,e, d'\,e'$ are both defined.

One can easily verify that, for all $d, d' \in [\![\rho_1 \to \cdots \to \rho_n \to o]\!]^B$, $e_1, e'_1 \in [\![\rho_1]\!]^B$, ..., $e_n, e'_n \in [\![\rho_n]\!]^B$, if $d \cong_{val,\rho_1\to\cdots\to\rho_n\to o} d'$, $e_1 \cong_{val,\rho_1} e'_1$, ..., $e_n \cong_{val,\rho_n} e'_n$ and $d\,e_1\,\cdots\,e_n, d'\,e'_1\,\cdots\,e'_n$ are both defined, then $val(d\,e_1\,\cdots\,e_n) = val(d'\,e'_1\,\cdots\,e'_n)$.

Generally, it is not guaranteed that such relations will be equivalence relations; rather they are partial equivalences (they are shown by Bezem [2] to be symmetric and transitive). Whether they are moreover reflexive, depends on the specific valuation function. The above discussion leads to the notion of *extensional interpretation*:

Definition 53. Let P be a program and let $\mathcal{I}$ be a Herbrand interpretation of P with valuation function $val_\mathcal{I}$. We say $\mathcal{I}$ is *extensional* if for all argument types ρ the relations $\cong_{val_\mathcal{I},\rho}$ are reflexive, i.e. for all $\mathsf{E} \in [\![\rho]\!]^B$, it holds that $\mathsf{E} \cong_{val_\mathcal{I},\rho} \mathsf{E}$.

In [3] it is shown that for a program P of $\mathcal{H}$ (actually, of the broader class of hoapata programs), $\mathcal{M}_{\mathsf{P}}$ is extensional in the sense of the above definition. Then, this property is used in order to collapse $\mathcal{M}_{\mathsf{P}}$ to an interpretation which is extensional in the usual set-theoretic sense (this construction goes beyond our scope and will not be discussed here).

Theorem 8 (Extensionality property)**.** *Let* P *be a program of the language* $\mathcal{H}$*. Then* $\mathcal{M}_{\mathsf{P}}$ *is extensional.*

Finally, we conclude this section with the monotonicity property of $\mathcal{M}_{\mathsf{P}}$, defined and proven in [4].

Definition 54. Let $\mathbb{S}$ be a functional type structure and val be a valuation function for $\mathbb{S}$. For every argument type ρ we define the relations $\leq^B_{val,\rho}$ on $[\![\rho]\!]^B$ as follows: Let $d, d' \in [\![\rho]\!]^B$; then $d \leq^B_{val,\rho} d'$ if and only if

1. $\rho = \iota$ and $d = d'$, or
2. $\rho = o$ and $val(d) \leq val(d')$, or
3. $\rho = \rho' \to \pi$ and $d\, e \leq^B_{val,\pi} d'\, e$ for all $e \in [\![\rho']\!]^B$, such that $d\, e, d'\, e$ are both defined.

Definition 55. Let P be a program, let $\mathcal{I}$ be a Herbrand interpretation of P with valuation function $val_{\mathcal{I}}$. We say $\mathcal{I}$ is *monotonic* if for all predicate types $\rho \to \pi$, E D $\leq^B_{val_{\mathcal{I}},\pi}$ E D′ holds for all E $\in [\![\rho \to \pi]\!]^B$ and all D, D′ $\in [\![\rho]\!]^B$ such that D $\leq^B_{val_{\mathcal{I}},\rho}$ D′.

Theorem 9 (Monotonicity property)**.** *Let* P *be a program of the language* $\mathcal{H}$*. Then* $\mathcal{M}_{\mathsf{P}}$ *is monotonic.*

The fact that $\mathcal{M}_{\mathsf{P}}$ enjoys the extensionality property ensures its suitability for providing extensional semantics for higher-order logic programs. Moreover, the minimum model property and the monotonicity property will play an important role for the developments of the next section.

4.4 Relationship between the two semantics

In this section we investigate the connections and the differences between the two semantic approaches presented in the previous sections.

Recall that the model $\mathcal{M}_{\mathsf{P}}$, which captures the intended meaning of a program P in the view of Bezem's semantics, by definition assigns to all ground atoms the same truth values as $\mathsf{M}_{\mathsf{Gr(P)}}$, i.e. the minimum model of the ground instantiation of the program. It is therefore justified that we restrict our attention to $\mathsf{M}_{\mathsf{Gr(P)}}$, instead of $\mathcal{M}_{\mathsf{P}}$, in our attempt to compare Bezem's semantics and the domain theoretic semantics of Section 4.2. As a result, we will not need to consider Bezem-type interpretations. In this section (and its Subsections 4.4.1 and 4.4.2), when we talk about "an interpretation (respectively, state) of program P" we will always mean "an interpretation (respectively, state) under the domain theoretic semantics"; similarly, any reference to "an interpretation of Gr(P)", will always mean "an interpretation of the form of Definition 4".

To help us transcend the differences between the two semantic approaches when comparing them, we introduce two key notions, namely that of the *ground restriction* of a

domain-theoretic interpretation and its complementary notion of the *semantic extension* of $\mathsf{M}_{\mathsf{Gr(P)}}$.

But first we present the following *Substitution Lemma*, which will be useful in the proofs of later results.

Lemma 7 (Substitution Lemma)**.** *Let* P *be a program and I be an interpretation of* P*. Also let* E *be an expression and* θ *be a ground substitution with* $vars(\mathsf{E}) \subseteq dom(\theta)$*. If s is a Herbrand state such that, for all* $\mathsf{V} \in vars(\mathsf{E})$*,* $s(\mathsf{V}) = [\![\theta(\mathsf{V})]\!]^W(I)$*, then* $[\![\mathsf{E}]\!]^W_s(I) = [\![\mathsf{E}\theta]\!]^W(I)$*.*

Proof. By a structural induction on E.

For the basis case, if $\mathsf{E} = \mathsf{p}$ or $\mathsf{E} = \mathsf{c}$ then the statement reduces to an identity and if $\mathsf{E} = \mathsf{V}$ then it holds by assumption.

For the induction step, we first examine the case that $\mathsf{E} = (\mathsf{f}\ \mathsf{E}_1\ \cdots\ \mathsf{E}_n)$; then $[\![\mathsf{E}]\!]^W_s(I) = I(\mathsf{f})\ [\![\mathsf{E}_1]\!]^W_s(I)\ \cdots\ [\![\mathsf{E}_n]\!]^W_s(I)$ and $[\![\mathsf{E}\theta]\!]^W(I) = I(\mathsf{f})\ [\![\mathsf{E}_1\theta]\!]^W(I)\ \cdots\ [\![\mathsf{E}_n\theta]\!]^W(I)$. By the induction hypothesis, $[\![\mathsf{E}_1]\!]^W_s(I) = [\![\mathsf{E}_1\theta]\!]^W(I), \ldots, [\![\mathsf{E}_n]\!]^W_s(I) = [\![\mathsf{E}_n\theta]\!]^W(I)$, thus we have $[\![\mathsf{E}]\!]^W_s(I) = [\![\mathsf{E}\theta]\!]^W(I)$.

Now consider the case that $\mathsf{E} = (\mathsf{E}_1\,\mathsf{E}_2)$. We have $[\![\mathsf{E}]\!]^W_s(I) = [\![\mathsf{E}_1]\!]^W_s(I)\,[\![\mathsf{E}_2]\!]^W_s(I)$ and $[\![\mathsf{E}\theta]\!]^W(I) = [\![\mathsf{E}_1\theta]\!]^W(I)\,[\![\mathsf{E}_2\theta]\!]^W(I)$. Again, applying the induction hypothesis, we conclude that $[\![\mathsf{E}]\!]^W_s(I) = [\![\mathsf{E}\theta]\!]^W(I)$.

Finally, if $\mathsf{E} = (\mathsf{E}_1 \approx \mathsf{E}_2)$ we have that $[\![\mathsf{E}]\!]^W_s(I) = T$ iff $[\![\mathsf{E}_1]\!]^W_s(I) = [\![\mathsf{E}_2]\!]^W_s(I)$, which, by the induction hypothesis, holds iff $[\![\mathsf{E}_1\theta]\!]^W(I) = [\![\mathsf{E}_2\theta]\!]^W(I)$. Moreover, we have $[\![\mathsf{E}\theta]\!]^W(I) = T$ iff $[\![\mathsf{E}_1\theta]\!]^W(I) = [\![\mathsf{E}_2\theta]\!]^W(I)$, therefore we conclude that $[\![\mathsf{E}]\!]^W_s(I) = T$ iff $[\![\mathsf{E}\theta]\!]^W(I) = T$. □

Given a Herbrand interpretation *I* of a program of $\mathcal{H}$, it is straightforward to devise a corresponding interpretation of the ground instantiation of the program, by restricting *I* to only assigning truth values to ground atoms. As expected, such a restriction of a model of the program produces a model of its ground instantiation. This idea is formalized in the following definition and theorem.

Definition 56. Let P be a program, *I* be a Herbrand interpretation of P and Gr(P) be the ground instantiation of P. We define the *ground restriction* of *I*, which we denote by $I|_{\mathsf{Gr(P)}}$, to be an interpretation of Gr(P), such that, for every $\mathsf{A} \in U_{\mathsf{P},o}$, $I|_{\mathsf{Gr(P)}}(\mathsf{A}) = [\![\mathsf{A}]\!]^W(I)$.

Theorem 10. *Let* P *be a program and* Gr(P) *be its ground instantiation. Also let M be a Herbrand model of* P *and* $M|_{\mathsf{Gr(P)}}$ *be the ground restriction of M. Then* $M|_{\mathsf{Gr(P)}}$ *is a model of* Gr(P)*.*

Proof. By definition, each clause in Gr(P) is of the form $\mathsf{p}\,\mathsf{E}_1\ \cdots\ \mathsf{E}_n \leftarrow \mathsf{B}_1\theta, \ldots, \mathsf{B}_k\theta$, i.e. the ground instance of a clause $\mathsf{p}\,\mathsf{V}_1\ \cdots\ \mathsf{V}_\mathsf{n} \leftarrow \mathsf{B}_1, \ldots, \mathsf{B}_k$ in P with respect to a ground substitution θ, such that $dom(\theta)$ includes $\mathsf{V}_1, \ldots, \mathsf{V}_n$ and all other variables appearing in the body of the clause and $\theta(\mathsf{V}_i) = \mathsf{E}_i$, for all $i \in \{1, \ldots, n\}$. Let *s* be a Herbrand state such that $s(\mathsf{V}) = [\![\theta(\mathsf{V})]\!]^W(M)$, for all $\mathsf{V} \in dom(\theta)$. By the Substitution Lemma (Lemma 7) and the definition of $M|_{\mathsf{Gr(P)}}$, $[\![\mathsf{p}\,\mathsf{V}_1\ \cdots\ \mathsf{V}_n]\!]^W_s(M) = [\![\mathsf{p}\,\mathsf{E}_1\ \cdots\ \mathsf{E}_n]\!]^W(M) = M|_{\mathsf{Gr(P)}}(\mathsf{p}\,\mathsf{E}_1\ \cdots\ \mathsf{E}_n)$. Similarly, for each atom B_i in the body of the clause, we have $[\![\mathsf{B}_i]\!]^W_s(M) = [\![\mathsf{B}_i\theta]\!]^W(M) = M|_{\mathsf{Gr(P)}}(\mathsf{B}_i\theta)$ by the same argument. For each B_i that is of the form $(\mathsf{E}_1 \approx \mathsf{E}_2)$, we have $[\![\mathsf{B}_i]\!]^W_s(M) = [\![\mathsf{B}_i\theta]\!]^W(M) = [\![(\mathsf{E}_1\theta \approx \mathsf{E}_2\theta)]\!]^W(M)$ by the Substitution Lemma and $[\![(\mathsf{E}_1\theta \approx \mathsf{E}_2\theta)]\!]^W(M) = T$ iff $\mathsf{E}_1\theta = \mathsf{E}_2\theta$; because $M|_{\mathsf{Gr(P)}}(\mathsf{B}_i\theta) = M|_{\mathsf{Gr(P)}}((\mathsf{E}_1\theta \approx \mathsf{E}_2\theta)) = T$, iff $\mathsf{E}_1\theta = \mathsf{E}_2\theta$, it follows again that $[\![\mathsf{B}_i]\!]^W_s(M) = M|_{\mathsf{Gr(P)}}(\mathsf{B}_i\theta)$. Consequently, if $M|_{\mathsf{Gr(P)}}(\mathsf{B}_i\theta) = T$ for

all $i \in \{1, \ldots, k\}$, we also have that $[\![\mathsf{B}_i]\!]^W_s(M) = T, 1 \leq i \leq k$. As M is a model of P, this implies that $[\![\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_n]\!]^W_s(M) = M|_{\mathsf{Gr(P)}}(\mathsf{p}\,\mathsf{E}_1 \cdots \mathsf{E}_n) = T$ and therefore $M|_{\mathsf{Gr(P)}}$ is a model of $\mathsf{Gr}(\mathsf{P})$. □

Corollary 1. *Let* P *be a program and* $\mathsf{Gr}(\mathsf{P})$ *be its ground instantiation. Then,* $\mathsf{M}_{\mathsf{Gr(P)}} \leq M_{\mathsf{P}}|_{\mathsf{Gr(P)}}$.

Proof. By Theorem 10, $M_{\mathsf{P}}|_{\mathsf{Gr(P)}}$ is a model of $\mathsf{Gr}(\mathsf{P})$. The corollary follows because $\mathsf{M}_{\mathsf{Gr(P)}}$ is the minimum model of $\mathsf{Gr}(\mathsf{P})$. □

Corollary 1, which is actually an alternative form of the minimum model property of $\mathcal{M}_{\mathsf{P}}$ (see Theorem 7), describes the one direction of the relation between the $\leq^W_{Int_{\mathsf{P}}}$-minimum Herbrand model of a program and the $\leq$-minimum model of its ground instantiation. In order to prove the equivalence of the two semantics under consideration, we also introduce the concept of *semantic extensions*.

Definition 57. Let P be a program and $\mathsf{M}_{\mathsf{Gr(P)}}$ be the $\leq$-minimum model of $\mathsf{Gr}(\mathsf{P})$. Let $\mathsf{E} \in U_{\mathsf{P},\rho}$ be a ground expression of an argument type ρ and d be an element of $[\![\rho]\!]^W$. We will say that d is a *semantic extension* of E and write $d \rhd_\rho \mathsf{E}$ if one of the following cases applies:

- $\rho = \iota$ and $d = \mathsf{E}$;
- $\rho = o$ and $d = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E})$;
- $\rho = \rho' \to \pi$ and for all $d' \in [\![\rho']\!]^W$ and $\mathsf{E}' \in U_{P,\rho'}$, such that $d' \rhd_{\rho'} \mathsf{E}'$, it holds that $d\,d' \rhd_\pi \mathsf{E}\,\mathsf{E}'$.

Also, we say that a Herbrand interpretation I of P is a semantic extension of $\mathsf{M}_{\mathsf{Gr(P)}}$ and write $I \rhd \mathsf{M}_{\mathsf{Gr(P)}}$, if for each predicate p of type π appearing in P it holds that $I(\mathsf{p}) \rhd_\pi \mathsf{p}$.

Compared to that of the ground restriction presented earlier, the notion of extending a syntactic object to the realm of semantic elements, is more complicated. In fact, even the existence of a semantic extension is not immediately obvious. The next lemma guarantees that not only can such an extension be constructed for any expression of the language, but it also has an interesting property of mirroring the ordering of semantic objects with respect to $\leq^W_\tau$ in a corresponding ordering of the expressions with respect to $\leq^B_{\mathsf{M}_{\mathsf{Gr(P)}},\tau}$.

Lemma 8. *Let* P *be a program,* $\mathsf{Gr}(\mathsf{P})$ *be its ground instantiation and* $\mathsf{M}_{\mathsf{Gr(P)}}$ *be the* $\leq$*-minimum model of* $\mathsf{Gr}(\mathsf{P})$*. For every argument type* ρ *and every ground term* $\mathsf{E} \in U_{\mathsf{P},\rho}$*:*

1. *There exists* $e \in [\![\rho]\!]^W$ *such that* $e \rhd_\rho \mathsf{E}$*.*
2. *For all* $e, e' \in [\![\rho]\!]^W$ *and all* $\mathsf{E}' \in U_{\mathsf{P},\rho}$*, if* $e \rhd_\rho \mathsf{E}$*,* $e' \rhd_\rho \mathsf{E}'$ *and* $e \leq^W_\rho e'$*, then* $\mathsf{E} \leq^B_{\mathsf{M}_{\mathsf{Gr(P)}},\rho} \mathsf{E}'$*.*

Proof. We prove both statements simultaneously, performing an induction on the structure of ρ. Specifically, the first statement is proven by showing that in each case we can construct a function e of type ρ, which is monotonic with respect to $\leq^W_\rho$ and satisfies $e \rhd_\rho \mathsf{E}$.

In the basis case, the construction of e for types ι and o is trivial. Also, if $\rho = \iota$, then both $\rhd_\rho$ and $\leq^W_\rho$ reduce to equality, so we have $\mathsf{E} = \mathsf{E}'$, which in this case is equivalent to $\mathsf{E} \leq^B_{\mathsf{M}_{\mathsf{Gr(P)}},\rho} \mathsf{E}'$. On the other hand, for $\rho = o$, $\rhd_\rho$ identifies with equality, while $\leq^W_\rho$ and $\leq^B_{\mathsf{M}_{\mathsf{Gr(P)}},\rho}$ identify with $\leq$, so we have that $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}) = e \leq e' = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}')$ implies $\mathsf{E} \leq^B_{\mathsf{M}_{\mathsf{Gr(P)}},\rho} \mathsf{E}'$.

For a more complex type $\rho = \rho_1 \to \cdots \to \rho_n \to o$, $n > 0$, we can easily construct e, as follows:

$$e\, e_1 \cdots e_n = \begin{cases} F, & \text{if there exist } d_1, \ldots, d_n \text{ and ground terms } \mathsf{D}_1, \ldots, \mathsf{D}_n \text{ such that,} \\ & \text{for all } i, e_i \leq^W_{\rho_i} d_i, d_i \rhd_{\rho_i} \mathsf{D}_i \text{ and } \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}\,\mathsf{D}_1 \cdots \mathsf{D}_n) = F \\ T, & \text{otherwise.} \end{cases}$$

To see that e is monotonic, consider $e_1, \ldots, e_n, e'_1, \ldots, e'_n$, such that $e'_1 \leq^W_{\rho_1} e_1, \ldots, e'_n \leq^W_{\rho_n} e_n$ and observe that $e\, e_1 \cdots e_n = F$ implies $e\, e'_1 \cdots e'_n = F$, due to the transitivity of $\leq^W_{\rho_i}$. We will now show that $e \rhd_\rho \mathsf{E}$, i.e. for all $e_1, \ldots, e_n$ and $\mathsf{E}_1, \ldots, \mathsf{E}_n$ such that $e_1 \rhd_{\rho_1} \mathsf{E}_1, \ldots, e_n \rhd_{\rho_n} \mathsf{E}_n$, it holds $e\, e_1 \cdots e_n = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}\,\mathsf{E}_1 \cdots \mathsf{E}_n)$. This is trivial if $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}\ \mathsf{E}_1 \cdots \mathsf{E}_n) = F$, since $e_i \leq^W_{\rho_i} e_i$. Consider the case that $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}\ \mathsf{E}_1 \cdots \mathsf{E}_n) = T$. For the sake of contradiction, assume $e\, e_1 \cdots e_n = F$. Then, by the construction of e, there must exist $d_1, \ldots, d_n$ and $\mathsf{D}_1, \ldots, \mathsf{D}_n$ such that, for all i, $e_i \leq^W_{\rho_i} d_i$, $d_i \rhd_{\rho_i} \mathsf{D}_i$ and $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}\,\mathsf{D}_1 \cdots \mathsf{D}_n) = F$. By the induction hypothesis, we have that $\mathsf{E}_i \leq^B_{\mathsf{M}_{\mathsf{Gr(P)}},\rho_i} \mathsf{D}_i$, for all $i \in \{1, \ldots, n\}$. This, by the monotonicity property of $\mathsf{M}_{\mathsf{Gr(P)}}$ (Theorem 9), yields that $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}\,\mathsf{E}_1 \cdots \mathsf{E}_n) = T \leq \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}\ \mathsf{D}_1 \cdots \mathsf{D}_n) = F$, which is obviously a contradiction. Therefore it has to be that $e\, e_1 \cdots e_n = T$.

Finally, in order to prove the second statement and conclude the induction step, we need to show that for all terms $\mathsf{D}_1 \in U_{\mathsf{P},\rho_1}, \ldots, \mathsf{D}_n \in U_{\mathsf{P},\rho_n}$, it holds $\mathsf{E}\ \mathsf{D}_1 \cdots \mathsf{D}_n \leq^B_{\mathsf{M}_{\mathsf{Gr(P)}},o} \mathsf{E}'\ \mathsf{D}_1 \cdots \mathsf{D}_n$. By the induction hypothesis, there exist $d_1, \ldots, d_n$, such that $d_1 \rhd_{\rho_1} \mathsf{D}_1, \ldots, d_n \rhd_{\rho_n} \mathsf{D}_n$. Because $e \rhd_\rho \mathsf{E}$ and $\mathsf{E}\ \mathsf{D}_1 \cdots \mathsf{D}_n$ is of type o, by definition we have that $e\ d_1 \cdots d_n = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}\ \mathsf{D}_1 \cdots \mathsf{D}_n)$. Similarly, we have $e'\ d_1 \cdots d_n = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{E}'\ \mathsf{D}_1 \cdots \mathsf{D}_n)$. Moreover, by $e \leq^W_\rho e'$ we have that $e\ d_1 \cdots d_n \leq^W_o e'\ d_1 \cdots d_n$. This yields the desired result, since $\leq^W_o$ identifies with $\leq^B_{\mathsf{M}_{\mathsf{Gr(P)}},o}$. □

The semantic extension constructed as part of the proof of the above lemma, is one of possibly many that an expression may possess. In fact, even for simple programs such as the one of the following example, an expression may often have more than one semantic extensions.

Example 10. Assume that $\mathtt{q} : \iota \to o$ and $\mathtt{r} : (\iota \to o) \to o$. Then the program:

```
q a ←
r Q ←
```

has the following ground instantiation:

```
q a ←
r q ←
```

The only semantic extension of q is the relation $\{\mathtt{a}\}$, however r has two semantic extensions, namely the relations $\{\{\mathtt{a}\}\}$ and $\{\{\}, \{\mathtt{a}\}\}$. □

The following variation of the Substitution Lemma states that if the building elements of an expression are assigned meanings that are semantic extensions of their syntactic counterparts, then the meaning of the expression is itself a semantic extension of the expression.

Lemma 9. *Let* P *be a program,* Gr(P) *be its ground instantiation and I be a Herbrand interpretation of* P*. Also, let* E *be an expression of some argument type* ρ *and let s be a Herbrand state and* θ *be a ground substitution, with domain* $vars(\mathsf{E})$*. If, for all predicates* p *of type* π *appearing in* E*,* $I(\mathsf{p}) \rhd_\pi \mathsf{p}$ *and, for all variables* V *of type* ρ' *in* $vars(\mathsf{E})$*,* $s(\mathsf{V}) \rhd_{\rho'} \theta(\mathsf{V})$*, then* $[\![\mathsf{E}]\!]^W_s(I) \rhd_\rho \mathsf{E}\theta$*.*

Proof. The proof is by induction on the structure of E. The basis cases $\mathsf{E} = \mathsf{p}$ and $\mathsf{E} = \mathsf{V}$ hold by assumption and $\mathsf{E} = c : \iota$ is trivial. For the first case of the induction step, let $\mathsf{E} = (\mathsf{f}\ \mathsf{E}_1\ \cdots\ \mathsf{E}_n)$, where $\mathsf{E}_1, \ldots, \mathsf{E}_n$ are of type ι. By the induction hypothesis, we have that $[\![\mathsf{E}_1]\!]^W_s(I) \rhd_\iota \mathsf{E}_1\theta, \ldots, [\![\mathsf{E}_n]\!]^W_s(I) \rhd_\iota \mathsf{E}_n\theta$. As $\rhd_\iota$ is defined as equality, we have that $[\![\mathsf{E}]\!]^W_s(I) = I(\mathsf{f})\ [\![\mathsf{E}_1]\!]^W_s(I)\ \cdots\ [\![\mathsf{E}_n]\!]^W_s(I) = \mathsf{f}\ \mathsf{E}_1\theta\ \cdots\ \mathsf{E}_n\theta = \mathsf{E}\theta$ and therefore $[\![\mathsf{E}]\!]^W_s(I) \rhd_\iota \mathsf{E}\theta$. For the second case, let $\mathsf{E} = (\mathsf{E}_1\ \mathsf{E}_2)$, where E_1 is of type $\rho_1 = \rho_2 \to \pi$ and E_2 is of type ρ_2; then, $[\![\mathsf{E}]\!]^W_s(I) = [\![\mathsf{E}_1]\!]^W_s(I)\ [\![\mathsf{E}_2]\!]^W_s(I)$. By the induction hypothesis, $[\![\mathsf{E}_1]\!]^W_s(I) \rhd_{\rho_2\to\pi} \mathsf{E}_1\theta$ and $[\![\mathsf{E}_2]\!]^W_s(I) \rhd_{\rho_2} \mathsf{E}_2\theta$, thus, by definition, $[\![\mathsf{E}]\!]^W_s(I) = [\![\mathsf{E}_1]\!]^W_s(I)\ [\![\mathsf{E}_2]\!]^W_s(I) \rhd_\pi \mathsf{E}_1\theta\ \mathsf{E}_2\theta = (\mathsf{E}_1\ \mathsf{E}_2)\theta = \mathsf{E}\theta$. Finally, we have the case that $\mathsf{E} = (\mathsf{E}_1 \approx \mathsf{E}_2)$, where E_1 and E_2 are both of type ι. The induction hypothesis yields $[\![\mathsf{E}_1]\!]^W_s(I) \rhd_\iota \mathsf{E}_1\theta$ and $[\![\mathsf{E}_2]\!]^W_s(I) \rhd_\iota \mathsf{E}_2\theta$ or, since $\rhd_\iota$ is defined as equality, $[\![\mathsf{E}_1]\!]^W_s(I) = \mathsf{E}_1\theta$ and $[\![\mathsf{E}_2]\!]^W_s(I) = \mathsf{E}_2\theta$. Then $[\![\mathsf{E}_1]\!]^W_s(I) = [\![\mathsf{E}_2]\!]^W_s(I)$ iff $\mathsf{E}_1\theta = \mathsf{E}_2\theta$ and, equivalently, $[\![\mathsf{E}]\!]^W_s(I) = T$ iff $\mathsf{E}\theta = T$, which implies $[\![\mathsf{E}]\!]^W_s(I) \rhd_o \mathsf{E}\theta$. □

The above lemma verifies an anticipated property of the semantic extensions of $\mathsf{M}_{\mathsf{Gr(P)}}$, made explicit in the next corollary.

Corollary 2. *Let* P *be a program,* Gr(P) *be its ground instantiation and I be a Herbrand interpretation of* P*. Let* $\mathsf{M}_{\mathsf{Gr(P)}}$ *be the* $\leq$*-minimum model of* Gr(P)*. If* $I \rhd \mathsf{M}_{\mathsf{Gr(P)}}$ *then* $I|_{\mathsf{Gr(P)}} = \mathsf{M}_{\mathsf{Gr(P)}}$*.*

Proof. We need to show that, for any ground atom $\mathsf{A} \in U_{\mathsf{P},o}$, $I|_{\mathsf{Gr(P)}}(\mathsf{A}) = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{A})$, which, by definition, is equivalent to $[\![\mathsf{A}]\!]^W(I) = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{A})$. The latter is immediate from the above lemma, as $I \rhd \mathsf{M}_{\mathsf{Gr(P)}}$ implies that $I(\mathsf{p}) \rhd_\pi \mathsf{p}$ for every predicate p of every type π and, A being ground, we can take θ to be empty. □

Maybe a little less apparent is the fact that the converse is not necessarily true; that is, an interpretation whose ground restriction coincides with $\mathsf{M}_{\mathsf{Gr(P)}}$, is not always a semantic extension of $\mathsf{M}_{\mathsf{Gr(P)}}$. This is demonstrated in the following example.

Example 11. Consider again the program of Example 10 and suppose we augment it by adding one more clause, as follows:

$$\begin{aligned} &\mathtt{q\ a} \leftarrow \\ &\mathtt{r\ Q} \leftarrow \\ &\mathtt{p\ R} \leftarrow \end{aligned}$$

where $\mathtt{p} : ((\iota \to o) \to o) \to o$. Then the ground instantiation of the program takes the following form:

$$\begin{aligned} &\mathtt{q\ a} \leftarrow \\ &\mathtt{r\ q} \leftarrow \\ &\mathtt{p\ r} \leftarrow \end{aligned}$$

The semantic extensions of $\mathtt{q}$ and $\mathtt{r}$ are as in Example 10. Let e be the relation $\{\{\}, \{\mathtt{a}\}\}$ and I be an interpretation of the higher-order program such that $I(\mathtt{q}) = \{\mathtt{a}\}$, $I(\mathtt{r}) = e$ and $I(\mathtt{p}) = \{e\}$. It is easy to check that I is a legitimate interpretation, i.e. it assigns monotonic relations to both r and p, and that $I|_{\mathsf{Gr(P)}} = \mathsf{M}_{\mathsf{Gr(P)}}$. Because $\mathsf{M}_{\mathsf{Gr(P)}}(\mathtt{pr})$ is true, every semantic extension of $\mathtt{p}$ should be true of all semantic extensions of $\mathtt{r}$, however this is not the case for $I(\mathtt{p})$. Indeed, even though the relation $d = \{\{\mathtt{a}\}\}$ is a semantic extension of $\mathtt{r}$, $I(\mathtt{p})\, d$ evaluates to false. Therefore I is not a semantic extension of $\mathsf{M}_{\mathsf{Gr(P)}}$. □

4.4.1 Equivalence of the two Semantics for Definitional Programs

In this section we demonstrate that the domain theoretic semantics of [31, 9] and Bezem's semantics of [2, 3] are equivalent for definitional programs.

The next theorem establishes the equivalence of the two semantics under consideration, in stating that their respective minimum models assign the same meaning to all ground atoms.

Theorem 11. *Let* P *be a definitional program and let* Gr(P) *be its ground instantiation. Let* M_{P} *be the* $\leq^W_{Int_{\mathsf{P}}}$*-minimum Herbrand model of* P *and let* $\mathsf{M}_{\mathsf{Gr(P)}}$ *be the minimum model of* Gr(P)*. Then, for every* $\mathsf{A} \in U_{\mathsf{P},o}$ *it holds* $[\![\mathsf{A}]\!]^W(M_{\mathsf{P}}) = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{A})$.

Proof. We will construct an interpretation N for P and prove some key properties for this interpretation. Then we will utilize these properties to prove the desired result. The definition of N is as follows:

For every $\mathsf{p} : \pi$ where π is of the form $\rho_1 \to \cdots \to \rho_n \to o$ and all $d_1 \in [\![\rho_1]\!]^W, \ldots, d_n \in [\![\rho_n]\!]^W$,

$$N(\mathsf{p})\, d_1 \cdots d_n = \begin{cases} F, & \text{if there exist } e_1, \ldots, e_n \text{ and ground terms } \mathsf{E}_1, \ldots, \mathsf{E}_n \text{ such that,} \\ & \text{for all } i, d_i \leq^W_{\rho_i} e_i, e_i \rhd_{\rho_i} \mathsf{E}_i \text{ and } \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\,\mathsf{E}_1 \cdots \mathsf{E}_n) = F \\ T, & \text{otherwise} \end{cases}$$

Observe that the above construction is the same as the one used in the proof of Lemma 8 (in the special case that the expression E of Lemma 8 consists of a single predicate constant p). Recall that this function was shown to be monotonic with respect to $\leq^W_\pi$ and a semantic extension of the corresponding expression; in this case p. Therefore N is a valid Herbrand interpretation of P and, moreover, $N \rhd \mathsf{M}_{\mathsf{Gr(P)}}$.

Next we prove that N is a model of P. Let $\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_n \leftarrow \mathsf{B}_1, \ldots, \mathsf{B}_k$ be a clause in P and let $\{\mathsf{V}_1, \ldots, \mathsf{V}_n, \mathsf{X}_1, \ldots, \mathsf{X}_m\}$, with $\mathsf{V}_i : \rho_i$, for all $i \in \{1, \ldots, n\}$, and $\mathsf{X}_i : \iota$, for all $i \in \{1, \ldots, m\}$, be the set of variables appearing in the clause. Then, it suffices to show that, for any tuple $(d_1, \ldots, d_n)$ of arguments and any Herbrand state s such that $s(\mathsf{V}_i) = d_i$ for all $i \in \{1, \ldots, n\}$, $N(\mathsf{p})\, d_1 \cdots d_n = F$ implies that, for at least one $j \in \{1, \ldots, k\}$, $[\![\mathsf{B}_j]\!]^W_s(N) = F$. Again, by the definition of N, we see that if $N(\mathsf{p})\, d_1 \cdots d_n = F$, then there exist $e_1, \ldots, e_n$ and ground terms $\mathsf{E}_1, \ldots, \mathsf{E}_n$ such that $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\,\mathsf{E}_1 \cdots \mathsf{E}_n) = F$, $d_1 \leq^W_{\rho_1} e_1, \ldots, d_n \leq^W_{\rho_n} e_n$ and $e_1 \rhd_{\rho_1} \mathsf{E}_1, \ldots, e_n \rhd_{\rho_n} \mathsf{E}_n$. Let θ be a ground substitution such that $\theta(\mathsf{V}_i) = \mathsf{E}_i$ for all $i \in \{1, \ldots, n\}$ and, for all $i \in \{1, \ldots, m\}$, $\theta(\mathsf{X}_i) = s(\mathsf{X}_i)$; then there exists a ground instance $\mathsf{p}\,\mathsf{E}_1 \cdots \mathsf{E}_n \leftarrow \mathsf{B}_1\theta, \ldots, \mathsf{B}_k\theta$ of the above clause in Gr(P). As $\mathsf{M}_{\mathsf{Gr(P)}}$ is a model of the ground program, $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\,\mathsf{E}_1 \cdots \mathsf{E}_n) = F$ implies that there exists at least one $j \in \{1, \ldots, k\}$ such that $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{B}_j\theta) = F$. We are going to show that the latter implies that $[\![\mathsf{B}_j]\!]^W_s(N) = F$, which proves that N is a model of P. Indeed, let s' be a Herbrand

state such that $s'(\mathsf{V}_i) = e_i \rhd_{\rho_i} \theta(\mathsf{V}_i) = \mathsf{E}_i$ for all $i \in \{1, \ldots, n\}$ and $s'(\mathsf{X}_i) = \theta(\mathsf{X}_i) = s(\mathsf{X}_i)$ for all $i \in \{1, \ldots, m\}$. As we have shown earlier, $N(\mathsf{p}') \rhd_{\pi'} \mathsf{p}'$ for any predicate $\mathsf{p}' : \pi'$, thus by Lemma 9 we get $[\![\mathsf{B}_j]\!]^W_{s'}(N) \rhd_o \mathsf{B}_j\theta$. Since B_j is of type o, the latter reduces to $[\![\mathsf{B}_j]\!]^W_{s'}(N) = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{B}_j\theta) = F$. Also, because $d_i \leq^W_{\rho_i} e_i$, i.e. $s \leq^W_{S_\mathsf{P}} s'$, by the second part of Lemma 6 we get $[\![\mathsf{B}_j]\!]^W_{s}(N) \leq^W_o [\![\mathsf{B}_j]\!]^W_{s'}(N)$, which makes $[\![\mathsf{B}_j]\!]^W_{s}(N) = F$.

Now we can proceed to prove that, for all $\mathsf{A} \in U_{\mathsf{P},o}$, $[\![\mathsf{A}]\!]^W(M_\mathsf{P}) = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{A})$. Let A be of the form $\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n$, where $\mathsf{p} : \rho_1 \to \cdots \to \rho_n \to o \in \mathsf{P}$ and let $d_1 = [\![\mathsf{E}_1]\!]^W(M_\mathsf{P}), \ldots, d_n = [\![\mathsf{E}_n]\!]^W(M_\mathsf{P})$. As we have shown, N is a Herbrand model of P, while M_P is the minimum, with respect to $\leq^W_{Int_\mathsf{P}}$, of all Herbrand models of P, therefore we have that $M_\mathsf{P} \leq^W_{Int_\mathsf{P}} N$. By definition, this gives us that $M_\mathsf{P}(\mathsf{p})\ d_1 \cdots d_n \leq^W_o N(\mathsf{p})\ d_1 \cdots d_n\ (1)$ and, by the first part of Lemma 6, that $d_1 \leq^W_{\rho_1} [\![\mathsf{E}_1]\!]^W(N), \ldots, d_n \leq^W_{\rho_n} [\![\mathsf{E}_n]\!]^W(N)\ (2)$. Moreover, for all predicates $\mathsf{p}' : \pi'$ in P, we have $N(\mathsf{p}') \rhd_{\pi'} \mathsf{p}'$ and thus, by Lemma 9, taking s and θ to be empty, we get $[\![\mathsf{E}_i]\!]^W(N) \rhd_{\rho_i} \mathsf{E}_i, 1 \leq i \leq n$. In conjunction with (2), the latter implies that if $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n) = F$ then $N(\mathsf{p})\ d_1 \cdots d_n = F$, or, in other words, that $N(\mathsf{p})\ d_1 \cdots d_n \leq \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n)$. Because of (1) and the fact that $\leq^W_o$ identifies with $\leq$, this makes it that $M_\mathsf{P}(\mathsf{p})\ d_1 \cdots d_n \leq \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n)\ (3)$. On the other hand, by Corollary 1, $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n) \leq M_\mathsf{P}|_{\mathsf{Gr(P)}}(\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n)$. By the definition of $M_\mathsf{P}|_{\mathsf{Gr(P)}}$ and the meaning of application, the latter becomes $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n) \leq M_\mathsf{P}|_{\mathsf{Gr(P)}}(\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n) = [\![\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n]\!]^W(M_\mathsf{P}) = M_\mathsf{P}(\mathsf{p})\ [\![\mathsf{E}_1]\!]^W(M_\mathsf{P}) \cdots [\![\mathsf{E}_n]\!]^W(M_\mathsf{P}) = M_\mathsf{P}(\mathsf{p})\ d_1 \cdots d_n$. The last relation and (3) can only be true simultaneously, if all the above relations hold as equalities, in particular if $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n) = [\![\mathsf{p}\ \mathsf{E}_1 \cdots \mathsf{E}_n]\!]^W(M_\mathsf{P})$. □

The importance of the above theorem lies in the fact that it demonstrates that for a significant class of higher-order programs, two different semantic approaches coincide. We feel that this result strengthens the importance of definitional programs, and it also suggests that the two alternative semantic approaches are useful tools for the further study of higher-order logic programs.

4.4.2 Programs with Existential Predicate Variables

We have considered the two existing extensional approaches to the semantics of higher-order logic programming and demonstrated that they coincide for the class of definitional programs. In this section we show that, if we allow higher-order predicate variables that are not formal parameters of a clause, to appear in its body, then the two semantic approaches do not continue to coincide.

Example 12. Consider the following (non definitional) $\mathcal{H}$ program:

$$\mathtt{p\ a} \leftarrow \mathtt{Q\ a}$$

Under Bezem's semantics, the above program intuitively states that $\mathtt{p}$ is true of $\mathtt{a}$ if there exists *a predicate that is defined in the program* that is true of $\mathtt{a}$. Following Bezem's semantics, we initially take the ground instantiation of the program, namely:

$$\mathtt{p\ a} \leftarrow \mathtt{p\ a}$$

and then compute the least model of the above program which assigns to the atom $\mathtt{p\ a}$ the value F.

Under the domain theoretic semantics, the atom $\mathtt{p\ a}$ has the value T in the minimum Herbrand model of the initial program. This is because in this semantics, our initial program

reads (intuitively speaking) as follows: "$\mathtt{p\ a}$ is true if there exists a relation that is true of $\mathtt{a}$"; actually, there exists one such relation, namely the set $\{\mathtt{a}\}$. This discrepancy between the domain theoretic semantics and Bezem's semantics is due to the fact that the former is based on *sets* and not on the syntactic entities that appear in the program. □

The discussion in the above example leads to the following lemma:

Lemma 10. *For $\mathcal{H}$ programs, Bezem's semantics and the domain theoretic semantics do not in general coincide.*

This implies that Theorem 11 fails for non definitional programs: in Example 12 notice that $[\![\mathtt{p\ a}]\!]^W(M_{\mathsf{P}}) \neq \mathsf{M}_{\mathsf{Gr(P)}}(\mathtt{p\ a})$. It comes as a natural question whether there exist any nontrivial sufficient conditions under which the two semantics coincide even for non definitional programs. An examination of Example 12 reveals that the discrepancy between the two semantics is due to the fact that there does not exist in the program any predicate which is true of the term "$\mathtt{a}$". More generally, atoms in the bodies of clauses that start with existentially quantified predicate variables (such as $\mathtt{Q}$ in Example 12) are (intuitively speaking) possible to lead to failure during execution of a program under Bezem's semantics; this is due to the fact that there does not always exist a predicate *defined in the program* that can take the place of such existential variables and succeed.

The above discussion suggests that a possible idea which would bring closer the domain theoretic and Bezem's semantics for $\mathcal{H}$ programs, would be to insist that our programs contain a *top predicate* for every possible type.[4] The addition of all such predicates in the program guarantees that any body atoms that start with existentially quantified variables, will never fail during execution. As it turns out, under this assumption the two semantics coincide again (see Theorem 12 below). However, the arguably strong quality of this assumption highlights an important difference rather than signifies a convergence between Bezem's and the domain theoretic semantics in the case of $\mathcal{H}$ programs.

Given any type $\pi = \rho_1 \to \cdots \to \rho_n \to o$, a predicate $\mathtt{top}_\pi$ can be defined by a fact:

$$\mathtt{top}_\pi\ \mathtt{V}_1 \cdots \mathtt{V}_n \leftarrow$$

where $\mathtt{V}_1, \ldots, \mathtt{V}_n$ are distinct variables of types $\rho_1, \ldots, \rho_n$ respectively. We then have the following theorem:

Theorem 12. *Let* P *be a program of $\mathcal{H}$ that contains a $\mathtt{top}_\pi$ predicate for every predicate type π. Let* Gr(P) *be the ground instantiation of* P*, let M_{P} be the $\sqsubseteq_{\mathcal{I}_{\mathsf{P}}}$-minimum Herbrand model of* P *and let* $\mathsf{M}_{\mathsf{Gr(P)}}$ *be the $\leq$-minimum model of* Gr(P)*. Then, for every* $\mathsf{A} \in U_{\mathsf{P},o}$ *it holds* $[\![\mathsf{A}]\!]^W(M_{\mathsf{P}}) = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{A})$*.*

Proof. In exactly the same way as in the proof of Theorem 11, we can construct an interpretation N for P and we can demonstrate that $N \rhd \mathsf{M}_{\mathsf{Gr(P)}}$. It suffices to show that N is a model of P. The proof of this fact differs from the one given for Theorem 11 due to the possible existence of predicate variables in the bodies of clauses that do not necessarily appear in the heads of the clauses.

[4] Such top relations are also used in λProlog in order to answer queries with uninstantiated predicate variables (see the relevant discussion in [26][page 50]).

Let $\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_n \leftarrow \mathsf{B}_1, \ldots, \mathsf{B}_k$ be a clause in P and let $\{\mathsf{V}_1, \ldots, \mathsf{V}_n, \mathsf{V}'_1, \ldots, \mathsf{V}'_m\}$ be the set of variables of the clause, where $\mathsf{V}_i : \rho_i$, for all $i \in \{1, \ldots, n\}$, are the formal parameters of the clause, and $\mathsf{V}'_i : \rho'_i$, for all $i \in \{1, \ldots, m\}$, are the variables that appear in the body of the clause but not in the head. Then, it suffices to show that, for any tuple $(d_1, \ldots, d_n)$ of arguments and any Herbrand state s such that $s(\mathsf{V}_i) = d_i$ for all $i \in \{1, \ldots, n\}$, $N(\mathsf{p})\, d_1 \cdots d_n = F$ implies that, for at least one $j \in \{1, \ldots, k\}$, $[\![\mathsf{B}_j]\!]^W_s(N) = F$. By the definition of N, we see that if $N(\mathsf{p})\, d_1 \cdots d_n = F$, there exist $e_1, \ldots, e_n$ and ground terms $\mathsf{E}_1, \ldots, \mathsf{E}_n$ such that $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\,\mathsf{E}_1 \cdots \mathsf{E}_n) = F$, $d_1 \sqsubseteq_{\rho_1} e_1, \ldots, d_n \sqsubseteq_{\rho_n} e_n$ and $e_1 \rhd_{\rho_1} \mathsf{E}_1, \ldots, e_n \rhd_{\rho_n} \mathsf{E}_n$. Let θ be a ground substitution such that $\theta(\mathsf{V}_i) = \mathsf{E}_i$ for all $i \in \{1, \ldots, n\}$; moreover, for all $i \in \{1, \ldots, m\}$, if $\mathsf{V}'_i : \iota$ then $\theta(\mathsf{V}'_i) = s(\mathsf{V}'_i)$ and if $\mathsf{V}'_i : \rho'_i \neq \iota$ then $\theta(\mathsf{V}'_i) = \mathtt{top}_{\rho'_i}$. Then there exists a ground instance $\mathsf{p}\,\mathsf{E}_1 \cdots \mathsf{E}_n \leftarrow \mathsf{B}_1\theta, \ldots, \mathsf{B}_k\theta$ of the above clause in $\mathsf{Gr(P)}$. As $\mathsf{M}_{\mathsf{Gr(P)}}$ is a model of $\mathsf{Gr(P)}$, $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{p}\,\mathsf{E}_1 \cdots \mathsf{E}_n) = F$ implies that there exists at least one $j \in \{1, \ldots, k\}$ such that $\mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{B}_j\theta) = F$. We are going to show that the latter implies that $[\![\mathsf{B}_j]\!]^W_s(N) = F$, which proves that N is a model of P. Indeed, let s' be a Herbrand state such that $s'(\mathsf{V}_i) = e_i$ for all $i \in \{1, \ldots, n\}$ and $s'(\mathsf{V}'_i) = [\![\theta(\mathsf{V}'_i)]\!]^W(N)$ for all $i \in \{1, \ldots, m\}$. Then, for all $i \in \{1, \ldots, n\}$, $s'(\mathsf{V}_i) \rhd_{\rho_i} \theta(\mathsf{V}_i) = \mathsf{E}_i$ and, for all $i \in \{1, \ldots, m\}$ such that V'_i is of type ι, $s'(\mathsf{V}'_i) = \theta(\mathsf{V}'_i) = s(\mathsf{V}'_i)$, which is equivalent to $s'(\mathsf{V}'_i) \rhd_\iota \theta(\mathsf{V}'_i)$. Also, since $N \rhd \mathsf{M}_{\mathsf{Gr(P)}}$, $N(\mathsf{p}') \rhd_{\pi'} \mathsf{p}'$ for any predicate $\mathsf{p}' : \pi'$ and this also implies that, for all $i \in \{1, \ldots, m\}$ such that V'_i is of some predicate type π, $s'(\mathsf{V}'_i) = N(\mathtt{top}_\pi) \rhd_\pi \theta(\mathsf{V}'_i) = \mathtt{top}_\pi$. By these remarks, Lemma 9 can be applied and yields that $[\![\mathsf{B}_j]\!]^W_{s'}(N) \rhd_o \mathsf{B}_j\theta$. Since B_j is of type o, the latter reduces to $[\![\mathsf{B}_j]\!]^W_{s'}(N) = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{B}_j\theta) = F$. Also, because $d_i \sqsubseteq_{\rho_i} e_i$, i.e. $s \sqsubseteq_{\mathcal{S}_\mathsf{P}} s'$, by the second part of Lemma 6 we get $[\![\mathsf{B}_j]\!]^W_s(N) \sqsubseteq_o [\![\mathsf{B}_j]\!]^W_{s'}(N)$, which makes $[\![\mathsf{B}_j]\!]^W_s(N) = F$.

Since N is a model of P, it is now straightforward to establish, in exactly the same way as in the proof of Theorem 11, that for all $\mathsf{A} \in U_{\mathsf{P},o}$, $[\![\mathsf{A}]\!]^W(M_\mathsf{P}) = \mathsf{M}_{\mathsf{Gr(P)}}(\mathsf{A})$. □

The above result suggests that we can retrieve an equivalence between Bezem's semantics and the domain theoretic semantics for all programs of the language $\mathcal{H}$, in the special case where $\mathtt{top}_\pi$ predicates exist for all types. We should however stress that this assumption would be of no consequence if the semantic approaches under consideration were not monotonic, as for example in the case that our source language allowed negation-as-failure.

5. GENERAL HIGHER-ORDER LOGIC PROGRAMS

In this chapter we consider the extension of our higher-order language $\mathcal{H}$ so as to include negation and we define the concepts of stratification and local stratification in the higher-order case. Moreover, we study different approaches to the semantics of such extended programs, which we will call *general* higher-order logic programs. Focusing on Bezem's semantics, we discuss different choices of valuation functions and whether each one enjoys the extensionality property. We conclude that only the infinite-valued approach is suitable for defining an extensional semantics for general higher-order programs when used in conjunction with Bezem's technique.

5.1 Syntax

We extend the syntax of the higher-order language $\mathcal{H}$ by adding the *negation* constant $\sim$ to the alphabet and augmenting the set of expressions of the language $\mathcal{H}$ with negated atoms:

Definition 58. If E is a term of $\mathcal{H}$ of type o then $(\sim \mathsf{E})$ is an expression of $\mathcal{H}$ of type o. Moreover, if θ is a substitution, then $(\sim \mathsf{E})\theta = (\sim \mathsf{E}\theta)$.

Expressions of type o will collectively be called *literals*. In particular, expressions of the form $(\sim \mathsf{E})$ will be called *negative literals*. Literals that do not contain negation, i.e. atoms and expressions of the form $(\mathsf{E}_1 \approx \mathsf{E}_2)$, will be called *positive literals*. A *ground literal* (respectively, *ground positive literal* or *ground negative literal*) is a literal (respectively, positive literal or negative literal) that contains no variables.

The next definition describes the form of the program clauses of our extended language.

Definition 59. A *general clause* of $\mathcal{H}$ is a formula $\mathsf{p}\ \mathsf{V}_1 \cdots \mathsf{V}_n \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_m$, where p is a predicate constant of type $\rho_1 \to \cdots \to \rho_n \to o$, $\mathsf{V}_1, \ldots, \mathsf{V}_n$ are distinct variables of types $\rho_1, \ldots, \rho_n$ respectively and $\mathsf{L}_1, \ldots, \mathsf{L}_m$ are literals. A *general program* P of $\mathcal{H}$ is a finite set of general clauses.

From now on, when we refer to a program (respectively, clause) of $\mathcal{H}$ we will always mean a general program (respectively, general clause) of the form of the above definition. So that no confusion arises, programs (respectively, clauses) that do not contain negation, as they were defined in Section 4.1, will be called *positive* programs (respectively, *positive* clauses).

Recall that for a positive program P, $U^+_{\mathsf{P},o}$, is defined as the set of all ground expressions of type o, that can be formed out of the individual constants, function symbols and predicate constants in the program. In the case of general programs, this definition also includes ground negative literals.

Definition 60. For a general program P, we define $U^+_{\mathsf{P},o}$ to be the set $U_{\mathsf{P},o} \cup \{(\mathsf{E}_1 \approx \mathsf{E}_2) \mid \mathsf{E}_1, \mathsf{E}_2 \in U_{\mathsf{P},\iota}\} \cup \{(\sim \mathsf{E}) \mid \mathsf{E} \in U_{\mathsf{P},o}\}$.

Example 13. The program below defines the `subset` relation over unary predicates:

```
subset S1 S2 ← ∼(nonsubset S1 S2)
nonsubset S1 S2 ← (S1 X), ∼(S2 X)
```

Given unary predicates `p` and `q`, `subset p q` is true iff `p` is a subset of `q`. □

Example 14. For a more "real-life" higher-order logic program with negation, assume that we have a unary predicate `movie M` and a binary predicate `ranking M R` which returns the ranking `R` of a given movie `M`. Consider also the following first-order predicate that defines a preference over movies based on their ranking:

```
prefer M1 M2 ← movie M1, movie M2, ranking M1 R1, ranking M2 R2, R1>R2
```

The following higher-order predicate `winnow` (see for example [13]) can be used to select all the "best" tuples `T` out of a given relation `R` based on a preference relation `P`:

```
winnow P R T ← R T, ∼(bypassed P R T)
bypassed P R T ← R T1, P T1 T
```

Intuitively, `winnow` returns all the tuples `T` of the relation `R` such that there does not exist any tuple `T1` in the relation `R` that is better from `T` with respect to the preference relation `P`. For example, if we ask the query `?- winnow prefer movie T.` we expect as answers all those movies that have the highest possible ranking. Notice that since `winnow` is a higher-order predicate, it can be invoked with different arguments; for example, it can be used to select out of a `book` relation, all those books that have the lowest possible price, or out of a `flight` relation all those flights that go to London, and so on. □

5.2 Extended Bezem's Semantics

In Chapter 4, we exhibited how Bezem's semantics makes use of the traditional least fixed-point semantics of positive first-order logic programs, in order to provide an extensional semantics for higher-order positive programs. In this section, we attempt to do the same for general higher-order programs, by combining Bezem's technique with each of the three approaches to the semantics of first-order programs with negation, presented in Chapter 3. As we are going to see, this is not such a straightforward task: only the infinite-valued semantics proves to be a viable option, as both the well-founded and the stable model semantics fail to preserve extensionality. However, the class of stratified programs is found to be a remarkable exception, since the well-founded model of every stratified higher-order program is extensional and, moreover, two-valued. Note that the domain-theoretic technique of Section 4.2 has also been applied to general programs under both the infinite-valued semantics [8] and, more recently, the well-founded semantics [10].

The key idea behind extending Bezem's semantics in order to apply to higher-order logic programs with negation, is straightforward to state: given such a program, we first take its ground instantiation. The resulting program is a (possibly infinite) propositional program with negation and any of the standard models that might capture the meaning of such programs, could be used as a valuation function defining a Herbrand interpretation for the initial program itself. Then we proceed to examine whether the resulting interpretation is an extensional model of the program.

In Section 4.3 we defined a two-valued valuation function as a function of type $[\![o]\!]^B \to \mathbb{V}^2$. Because each of the approaches that we will consider is based on a different set of truth values, we need to respectively consider valuation functions with different domains.

Definition 61. Let $\mathbb{S}$ be a functional type structure and $\mathbb{V}$ be a domain of truth values. Any function *val* : $[\![o]\!]^B \to \mathbb{V}$ will be called a *valuation function* for $\mathbb{S}$ over $\mathbb{V}$. Moreover, *val* is called:

- two-valued, if $\mathbb{V} = \mathbb{V}^2 = \{F, T\}$;
- three-valued, if $\mathbb{V} = \mathbb{V}^3 = \{F, 0, T\}$;
- infinite-valued, if $\mathbb{V} = \mathbb{V}^\infty$.

Similarly, we can extend the notion of Herbrand interpretation, by considering interpretations with different types of valuation functions. As in the case of positive programs, all the interpretations we consider are built upon the Herbrand type structure, for which $[\![o]\!]^B = U^+_{\mathsf{P},o}$. Again, any $\mathsf{E} \in U_{\mathsf{P},o\to\pi}$ is a partial function, because it does not map expressions of the form $(\mathsf{E}_1 \approx \mathsf{E}_2)$ nor of the form $(\sim\mathsf{E})$, while any $\mathsf{E} \in U_{\mathsf{P},\rho\to\pi}$ where $\rho \neq o$ represents a total function.

We will appropriately restrict the valuation functions of two-valued, three-valued and infinite-valued interpretations so that they conform to the way that negation is interpreted under the respective semantic approaches; in a way, we ensure that our interpretations are "consistent". Therefore, a Herbrand interpretation $\mathcal{I}$ for a general program P is called two-valued, if the valuation function $\mathit{val}_\mathcal{I}(\cdot)$ of $\mathcal{I}$ is two-valued and satisfies the following:

1. for all $\mathsf{E}_1, \mathsf{E}_2 \in U_{\mathsf{P},\iota}$, $\mathit{val}_\mathcal{I}((\mathsf{E}_1 \approx \mathsf{E}_2)) = \begin{cases} F, & \text{if } \mathsf{E}_1 \neq \mathsf{E}_2 \\ T, & \text{if } \mathsf{E}_1 = \mathsf{E}_2 \end{cases}$ and
2. for all $\mathsf{E} \in U_{\mathsf{P},o}$, $\mathit{val}_\mathcal{I}((\sim\mathsf{E})) = \begin{cases} F, & \text{if } \mathit{val}(\mathsf{E}) = T \\ T, & \text{if } \mathit{val}(\mathsf{E}) = F \end{cases}$.

On the other hand, $\mathcal{I}$ is called three-valued, if $\mathit{val}_\mathcal{I}(\cdot)$ is a three-valued valuation function which satisfies all of the above, and, for all $\mathsf{E} \in U_{\mathsf{P},o}$, if $\mathit{val}(\mathsf{E}) = 0$ then $\mathit{val}_\mathcal{I}((\sim\mathsf{E})) = 0$. Finally, $\mathcal{I}$ is called infinite-valued, if $\mathit{val}_\mathcal{I}(\cdot)$ is infinite-valued and satisfies:

1. for all $\mathsf{E}_1, \mathsf{E}_2 \in U_{\mathsf{P},\iota}$, $\mathit{val}_\mathcal{I}((\mathsf{E}_1 \approx \mathsf{E}_2)) = \begin{cases} \mathit{F_0}, & \text{if } \mathsf{E}_1 \neq \mathsf{E}_2 \\ \mathit{T_0}, & \text{if } \mathsf{E}_1 = \mathsf{E}_2 \end{cases}$ and
2. for all $\mathsf{E} \in U_{\mathsf{P},o}$, $\mathit{val}_\mathcal{I}((\sim\mathsf{E})) = \begin{cases} T_{\alpha+1}, & \text{if } \mathit{val}(\mathsf{E}) = F_\alpha \\ F_{\alpha+1}, & \text{if } \mathit{val}(\mathsf{E}) = T_\alpha \\ 0, & \text{if } \mathit{val}(\mathsf{E}) = 0 \end{cases}$.

It is straightforward to extend the semantic function $[\![\cdot]\!]^B$ so as to define the semantics of expressions of the form $(\sim\mathsf{E})$.

Definition 62. Let P be a program and let E be a term of type o. Also, let $\mathcal{I}$ be a Herbrand interpretation and s a Herbrand state of P. We define $[\![(\sim\mathsf{E})]\!]^B_s(\mathcal{I}) = (\sim[\![\mathsf{E}]\!]^B_s(\mathcal{I}))$.

Given that a truth ordering $\leq$ is defined on every one of the truth domains we consider, i.e. the two-valued, three-valued and infinite-valued domain, Definition 49 of a higher-order model can be applied to interpretations of general programs with two-valued, three-valued or infinite-valued valuation functions.

The same holds for Definition 51 of the ordering of interpretations. For simplicity, we write $\leq$ (instead of variations of $\leq^B_{\mathcal{I}_\mathsf{P}}$) for all the orderings on all sets of interpretations that we

consider (i.e., two-valued, three-valued and infinite-valued ones). The following theorem generalises Bezem's minimum model result (see Theorem 7). Again, for all $\mathsf{E}_1, \mathsf{E}_2 \in U_{\mathsf{P},\iota}$, $(\mathsf{E}_1 \approx \mathsf{E}_2)$ is equivalent to the constant *true* if $\mathsf{E}_1 = \mathsf{E}_2$, otherwise it is equivalent to the constant *false*.

Theorem 13. *Let* P *be a program and let* Gr(P) *be its ground instantiation. Also, let* M *be a two-valued (respectively, three-valued or infinite valued) interpretation of* Gr(P) *and let* $\mathcal{M}$ *be the two-valued (respectively, three-valued or infinite valued) Herbrand interpretation of* P*, such that* $val_{\mathcal{M}}(\mathsf{A}) = \mathsf{M}(\mathsf{A})$ *for every* $\mathsf{A} \in U_{\mathsf{P},o}$*. Then,* $\mathcal{M}$ *is a Herbrand model of* P *if and only if* M *is a model of* Gr(P)*. Moreover,* $\mathcal{M}$ *is minimal with respect to* $\leq$ *if and only if* M *is minimal.*

Proof. Let M and $\mathcal{M}$ be two-valued interpretations of P and Gr(P) respectively.

Step 1 M is a model of Gr(P) $\Rightarrow$ $\mathcal{M}$ is a Herbrand model of P: For every Herbrand state $\mathcal{s}$ of P there exists a ground substitution θ such that $\theta(\mathsf{V}) = \mathcal{s}(\mathsf{V})$, and therefore $\mathcal{s}(\mathsf{V}) = [\![\theta(\mathsf{V})]\!]^B_{\mathcal{s}'}(\mathcal{M})$, for all states $\mathcal{s}'$ and variables V in P. Also, for every clause $\mathsf{A} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_m$ in P there exists a respective ground instance $\mathsf{A}\theta \leftarrow \mathsf{L}_1\theta, \ldots, \mathsf{L}_m\theta$ in Gr(P). As M is a model of Gr(P), $\mathsf{M}(\mathsf{A}\theta) \geq min\{\mathsf{M}(\mathsf{L}_1\theta), \ldots, \mathsf{M}(\mathsf{L}_m\theta)\}$. By assumption, $val_{\mathcal{M}}(\mathsf{A}\theta) = \mathsf{M}(\mathsf{A}\theta)$ and $val_{\mathcal{M}}(\mathsf{L}_i\theta) = \mathsf{M}(\mathsf{L}_i\theta)$ for all $i \leq m$. Moreover, it is easy to see (by a trivial induction on the structure of the expression) that $\mathsf{A}\theta = [\![\mathsf{A}]\!]^B_{\mathcal{s}}(\mathcal{M})$, which implies that $val_{\mathcal{M}}(\mathsf{A}\theta) = val_{\mathcal{M}}([\![\mathsf{A}]\!]^B_{\mathcal{s}}(\mathcal{M}))$. Similarly, $val_{\mathcal{M}}(\mathsf{L}_i\theta) = val_{\mathcal{M}}([\![\mathsf{L}_i]\!]^B_{\mathcal{s}}(\mathcal{M}))$, for all $i \leq m$. Then it follows immediately that $val_{\mathcal{M}}([\![\mathsf{A}]\!]^B_{\mathcal{s}}(\mathcal{M})) \geq min\{val_{\mathcal{M}}([\![\mathsf{L}_1]\!]^B_{\mathcal{s}}(\mathcal{M})), \ldots, val_{\mathcal{M}}([\![\mathsf{L}_m]\!]^B_{\mathcal{s}}(\mathcal{M}))\}$ which implies that $\mathcal{M}$ is a model of P.

Step 2 $\mathcal{M}$ is a Herbrand model of P $\Rightarrow$ M is a model of Gr(P): Every clause in Gr(P) is a ground instance of a clause $\mathsf{A} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_m$ in P and is therefore of the form $\mathsf{A}\theta \leftarrow \mathsf{L}_1\theta, \ldots, \mathsf{L}_m\theta$ for some ground substitution θ. Consider a Herbrand state s, such that $s(\mathsf{V}) = \theta(\mathsf{V})$ for every variable V in P. Because $\mathcal{M}$ is a model of P, we have that $val_{\mathcal{M}}([\![\mathsf{A}]\!]^B_{\mathcal{s}}(\mathcal{M})) \geq min\{val_{\mathcal{M}}([\![\mathsf{L}_1]\!]^B_{\mathcal{s}}(\mathcal{M})), \ldots, val_{\mathcal{M}}([\![\mathsf{L}_m]\!]^B_{\mathcal{s}}(\mathcal{M}))\}$. Again, it is easy to see that $\mathsf{A}\theta = [\![\mathsf{A}]\!]^B_{\mathcal{s}}(\mathcal{M})$ and therefore $val_{\mathcal{M}}(\mathsf{A}\theta) = val_{\mathcal{M}}([\![\mathsf{A}]\!]^B_{\mathcal{s}}(\mathcal{M}))$. Similarly, $val_{\mathcal{M}}(\mathsf{L}_i\theta) = val_{\mathcal{M}}([\![\mathsf{L}_i]\!]^B_{\mathcal{s}}(\mathcal{M}))$ for all $i \leq m$. Additionally, $val_{\mathcal{M}}(\mathsf{A}\theta) = \mathsf{M}(\mathsf{A}\theta)$ and $val_{\mathcal{M}}(\mathsf{L}_i\theta) = \mathsf{M}(\mathsf{L}_i\theta)$ for all $i \leq m$, so $\mathsf{M}(\mathsf{A}\theta) \geq min\{\mathsf{M}(\mathsf{L}_1\theta), \ldots, \mathsf{M}(\mathsf{L}_m\theta)\}$, which implies that M is a model of Gr(P).

Step 3 M is minimal $\Rightarrow$ $\mathcal{M}$ is minimal: Assume there exists a model $\mathcal{N}$ of P, distinct from $\mathcal{M}$, such that $\mathcal{N} \leq \mathcal{M}$. Then we can construct an interpretation N for Gr(P) such that for every ground atom A, $\mathsf{N}(\mathsf{A}) = val_{\mathcal{N}}(\mathsf{A})$. It is obvious that $\mathsf{N} \leq \mathsf{M}$, since $\mathsf{N}(\mathsf{A}) = val_{\mathcal{N}}(\mathsf{A}) \leq val_{\mathcal{M}}(\mathsf{A}) = \mathsf{M}(\mathsf{A})$. Also, N is distinct from M, since $\mathsf{N}(\mathsf{B}) = val_{\mathcal{N}}(\mathsf{B}) \neq val_{\mathcal{M}}(\mathsf{B}) = \mathsf{M}(\mathsf{B})$ for at least one ground atom B. As we showed in Step 2, the fact that $\mathcal{N}$ is a model of P implies that N is a model of Gr(P), which is of course a contradiction, since M is a minimal model of Gr(P). Therefore, $\mathcal{M}$ must be a minimal model of P.

Step 4 $\mathcal{M}$ is minimal $\Rightarrow$ M is minimal: By the reverse of the argument used in Step 3: Assume there exists a model N of Gr(P), distinct from M, such that $\mathsf{N} \leq \mathsf{M}$. Then we can construct an interpretation $\mathcal{N}$ for P such that for every ground atom A, $\mathsf{N}(\mathsf{A}) = val_{\mathcal{N}}(\mathsf{A})$. It is obvious that $\mathcal{N} \leq \mathcal{M}$, since $val_{\mathcal{N}}(\mathsf{A}) = \mathsf{N}(\mathsf{A}) \leq \mathsf{M}(\mathsf{A}) = val_{\mathcal{M}}(\mathsf{A})$ Also, $\mathcal{N}$ is distinct from $\mathcal{M}$, since their valuation functions are distinct. As we showed in

Step 1, the fact that N is a model of Gr(P) implies that $\mathcal{N}$ is a model of P, which is of course a contradiction, since $\mathcal{M}$ is a minimal model of P. Therefore, M must be a minimal model of Gr(P).

For three-valued and infinite-valued interpretations, the proof is identical. □

However, the ordering of infinite-valued interpretations based on the truth ordering on $\mathbb{V}^\infty$ is of no consequence for our developments. Instead, it will be more useful to extend the more relevant $\sqsubseteq$ ordering of first-order infinite-valued interpretations. We adopt the notation $\mathcal{I} \parallel v$ from Section 3.5, to signify the set of atoms which are assigned a certain truth value $v \in \mathbb{V}^\infty$ by an infinite-valued Herbrand interpretation $\mathcal{I}$; that is, $\mathcal{I} \parallel v = \{\mathsf{A} \mid \mathsf{A} \in U_{\mathsf{P},o}$ and $\mathit{val}_{\mathcal{I}}(\mathsf{A}) = v\}$. Then the relations $\sqsubseteq_\alpha$, $\sqsubset_\alpha$, $=_\alpha$, $\sqsubseteq$ and $\sqsubset$ on infinite-valued Herbrand interpretations of a higher-order program can be defined in exactly the same manner as in Section 3.5.

Eventually, the following counterpart of the previous theorem is obtained:

Theorem 14. *Let* P *be a program and let* Gr(P) *be its ground instantiation. Also, let* M *be an infinite-valued interpretation of* Gr(P) *and let* $\mathcal{M}$ *be the infinite-valued Herbrand interpretation of* P*, such that* $\mathit{val}_{\mathcal{M}}(\mathsf{A}) = \mathsf{M}(\mathsf{A})$ *for every* $\mathsf{A} \in U_{\mathsf{P},o}$*. Then,* $\mathcal{M}$ *is the minimum, with respect to* $\sqsubseteq$*, model of* P *if and only if* M *is the minimum, with respect to* $\sqsubseteq$*, model of* Gr(P)*.*

Proof. By Theorem 13, we already have that M is a model of Gr(P) iff $\mathcal{M}$ is a model of P. It remains to show that M is minimum iff $\mathcal{M}$ is minimum.

Step 1 M is minimum $\Rightarrow$ $\mathcal{M}$ is minimum: Assume there exists a model $\mathcal{N}$ of the higher-order program P, distinct from $\mathcal{M}$, which does not satisfy $\mathcal{M} \sqsubset \mathcal{N}$. Then we can construct an interpretation N for Gr(P) such that for every ground atom A, $\mathsf{N}(\mathsf{A}) = \mathit{val}_{\mathcal{N}}(\mathsf{A})$. It is obvious that $\mathsf{M} \not\sqsubset \mathsf{N}$, since $\mathsf{N}(\mathsf{A}) = \mathit{val}_{\mathcal{N}}(\mathsf{A})$ and $\mathit{val}_{\mathcal{M}}(\mathsf{A}) = \mathsf{M}(\mathsf{A})$ for every ground atom A imply that $\mathsf{N} \parallel v = \mathcal{N} \parallel v$ and $\mathsf{M} \parallel v = \mathcal{M} \parallel v$ for every truth value $v \in \mathbb{V}^\infty$. Also, N is distinct from M, since $\mathsf{N}(\mathsf{B}) = \mathit{val}_{\mathcal{N}}(\mathsf{B}) \neq \mathit{val}_{\mathcal{M}}(\mathsf{B}) = \mathsf{M}(\mathsf{B})$ for at least one ground atom B. By Theorem 13, the fact that $\mathcal{N}$ is a model of P implies that N is a model of Gr(P), which is of course a contradiction, since M is the minimum model of Gr(P) with respect to $\sqsubseteq$. Therefore, $\mathcal{M}$ must be the minimum model of P.

Step 2 $\mathcal{M}$ is minimum $\Rightarrow$ M is minimum: By the reverse of the argument used in Step 1: Assume there exists a model N of Gr(P), distinct from M, which does not satisfy $\mathsf{M} \sqsubset \mathsf{N}$. Then we can construct an interpretation $\mathcal{N}$ for P such that for every ground atom A, $\mathsf{N}(\mathsf{A}) = \mathit{val}_{\mathcal{N}}(\mathsf{A})$. It is obvious that $\mathcal{M} \not\sqsubset \mathcal{N}$, since $\mathit{val}_{\mathcal{N}}(\mathsf{A}) = \mathsf{N}(\mathsf{A})$ and $\mathsf{M}(\mathsf{A}) = \mathit{val}_{\mathcal{M}}(\mathsf{A})$ Also, $\mathcal{N}$ is distinct from $\mathcal{M}$, since their valuation functions are distinct. By Theorem 13, the fact that N is a model of Gr(P) implies that $\mathcal{N}$ is a model of P, which is of course a contradiction, since $\mathcal{M}$ is a minimal model of P. Therefore, M must be a minimal model of Gr(P).

□

5.2.1 Stable Model Extension

In this section we consider the extension of Bezem's technique under the stable model semantics. We demonstrate that this extension does not lead to an extensional semantics for the programs of our source language $\mathcal{H}$, by discussing examples of programs whose stable models are not extensional.

The stable model semantics, in its original form introduced in [19], is applied on the ground instantiation of a given first-order logic program with negation, which is a possibly infinite propositional program. In this respect, the stable models of the ground instantiation of a given higher-order program (which is again a possibly infinite propositional program), can obviously be used as valuation functions for the program. Recall that stable models are always two-valued.

Definition 63. Let P be a program and let $\mathrm{Gr}(\mathrm{P})$ be the ground instantiation of P. Also, let M be a stable model of $\mathrm{Gr}(\mathrm{P})$ and let $\mathcal{M}$ be the two-valued Herbrand interpretation of P, such that $\mathit{val}_{\mathcal{M}}(\mathrm{A}) = \mathrm{M}(\mathrm{A})$ for every $\mathrm{A} \in U_{\mathrm{P},o}$. Then $\mathcal{M}$ is called a *stable model* of P.

Clearly, by Theorem 13, $\mathcal{M}$ is a two-valued minimal model of P.

To demonstrate the non-extensionality of the stable models approach in the case of higher-order programs, it suffices to find a program that produces non-extensional stable models. The following very simple example does exactly this.

Example 15. Consider the higher-order program:

```
r Q ← ∼(s Q)
s Q ← ∼(r Q)
q a ←
p a ←
```

where the predicate variable `Q` of the first and second clause is of type $\iota \to o$. We at first take the ground instantiation of the above program:

```
r p ← ∼(s p)
r q ← ∼(s q)
s p ← ∼(r p)
s q ← ∼(r q)
q a ←
p a ←
```

Consider now the interpretation $\mathrm{M} = \{\texttt{(p a)}, \texttt{(q a)}, \texttt{(s p)}, \texttt{(r q)}\}$. One can easily check that M is a model of the ground instantiation of the program. However, the respective higher-order model is not extensional: since `p` and `q` are extensionally equal, the atoms `(s q)` and `(r p)` should also belong to M in order to ensure extensionality. It remains to show that M is also a stable model. Consider the reduct of the above program based on M:

```
r q ←
s p ←
q a ←
p a ←
```

Obviously, the least model of the reduct is the interpretation M, and therefore the interpretation of the initial program, that has M as its valuation function, is a stable model of

that program. In other words, we have found a program with a non-extensional stable model. □

Continuing the discussion on the above example, one can easily verify that the above program also has two extensional stable models, namely the Herbrand interpretations with valuation functions $\mathsf{M}_1 = \{\texttt{(p a)}, \texttt{(q a)}, \texttt{(s p)}, \texttt{(s q)}\}$ and $\mathsf{M}_2 = \{\texttt{(p a)}, \texttt{(q a)}, \texttt{(r p)}, \texttt{(r q)}\}$. This creates the hope that we could somehow adapt the standard stable model construction procedure in order to produce only extensional stable models. The following example makes this hope vanish.

Example 16. Consider the program:

```
r Q ← ∼(s Q), ∼(r p)
s Q ← ∼(r Q), ∼(s q)
q a ←
p a ←
```

where, in the first two clauses, `Q` is of type $\iota \to o$. The ground instantiation of the program is the following:

```
r p ← ∼(s p), ∼(r p)
r q ← ∼(s q), ∼(r p)
s p ← ∼(r p), ∼(s q)
s q ← ∼(r q), ∼(s q)
q a ←
p a ←
```

This program has the non-extensional stable model with valuation function $\mathsf{M} = \{\texttt{(p a)}, \texttt{(q a)}, \texttt{(s p)}, \texttt{(r q)}\}$. However, it has *no* extensional stable models: there are four extensional interpretations that are potential candidates, namely the interpretations with valuation functions $\mathsf{M}_1 = \{\texttt{(p a)}, \texttt{(q a)}\}$, $\mathsf{M}_2 = \{\texttt{(p a)}, \texttt{(q a)}, \texttt{(r p)}, \texttt{(r q)}\}$, $\mathsf{M}_3 = \{\texttt{(p a)}, \texttt{(q a)}, \texttt{(s p)}, \texttt{(s q)}\}$, and $\mathsf{M}_4 = \{\texttt{(p a)}, \texttt{(q a)}, \texttt{(s p)}, \texttt{(s q)}, \texttt{(r p)}, \texttt{(r q)}\}$; one can easily verify that none of these interpretations is a stable model of the ground instantiation of the program. The conclusion is that there exist general higher-order logic programs which have only non-extensional stable models. □

The above examples seem to suggest that the extensional approach of [2, 3] is incompatible with the stable model semantics.

5.2.2 Well-founded Extension

In this section we demonstrate that utilizing the well-founded model [18] of their ground instantiation as the valuation function, also fails to provide extensional models for our higher-order programs.

Definition 64. Let P be a program and let $\mathsf{Gr}(\mathsf{P})$ be the ground instantiation of P. Also, let $\mathsf{M}^{WF}_{\mathsf{Gr}(\mathsf{P})}$ be the well-founded model of $\mathsf{Gr}(\mathsf{P})$. We define $\mathcal{M}^{WF}_{\mathsf{P}}$ to be the three-valued Herbrand interpretation of P such that $\mathit{val}_{\mathcal{M}^{WF}_{\mathsf{P}}}(\mathsf{A}) = \mathsf{M}^{WF}_{\mathsf{Gr}(\mathsf{P})}(\mathsf{A})$ for every $\mathsf{A} \in U_{\mathsf{P},o}$. We call $\mathcal{M}^{WF}_{\mathsf{P}}$ the *well-founded model* of P.

By Theorem 13, $\mathcal{M}^{WF}_{\mathsf{P}}$ is a three-valued minimal model of P. However, as the following example proves, it is not always extensional.

Example 17. Consider the higher-order program P:

$$\begin{aligned}
&\texttt{s Q} \leftarrow \texttt{Q (s Q)}\\
&\texttt{p R} \leftarrow \texttt{R}\\
&\texttt{q R} \leftarrow \sim\texttt{(w R)}\\
&\texttt{w R} \leftarrow \sim\texttt{R}
\end{aligned}$$

where the predicate variable Q is of type $o \to o$ and the predicate variable R is of type o. Before stating formally the non-extensionality result, certain explanations at an intuitive level are in order. Consider first the predicate p of type $o \to o$. One can view p as representing the identity relation on truth values, i.e., as the relation $\{(v, v) \mid v \in \{F, 0, T\}\}$. It is not hard to see that the predicate q of type $o \to o$, represents exactly the same relation. However, the definition of q involves two applications of negation, while p is defined directly (without the use of negation).

Consider now the predicate s of type $(o \to o) \to o$ which can take as a parameter either p or q. When s takes p as a parameter, we get the following two clauses (by substituting p for Q and (s p) for R in the above program):

$$\begin{aligned}
&\texttt{s p} \leftarrow \texttt{p (s p)}\\
&\texttt{p (s p)} \leftarrow \texttt{(s p)}
\end{aligned}$$

A recursive definition of this form assigns to (s p), under the well-founded semantics, the value F. Consider on the other hand the case where s takes q as a parameter. Then, by doing analogous substitutions, we get the following three clauses:

$$\begin{aligned}
&\texttt{s q} \leftarrow \texttt{q (s q)}\\
&\texttt{q (s q)} \leftarrow \sim\texttt{(w (s q))}\\
&\texttt{w (s q)} \leftarrow \sim\texttt{(s q)}
\end{aligned}$$

Under the well-founded semantics, (s q) is assigned the value 0. In other words, despite the fact that p and q are extensionally equal (see also below), (s p) and (s q) have different truth values. In conclusion, the adaptation of the well-founded semantics under Bezem's technique does not lead to an extensional model in all cases. □

Of course, the above discussion is based on intuitive arguments, but it is not hard to formalize it. The main difficulty lies in establishing that p and q are extensionally equal because the above program has an infinite ground instantiation Gr(P) (see the proof of Lemma 11 that follows). The following lemma suggests that the well-founded model $\mathcal{M}_{\mathsf{P}}^{WF}$ of our example program P is not extensional.

Lemma 11. *The well-founded model $\mathcal{M}_{\mathsf{P}}^{WF}$ of the program of Example 17 is not extensional.*

Proof. Recall that the predicate variable Q is of type $o \to o$ and the predicate variable R

is of type o. The ground instantiation of the program is infinite, as so:

```
s p ← p (s p)
s q ← q (s q)
s w ← w (s w)
p (s p) ← (s p)
p (s q) ← (s q)
p (s w) ← (s w)
q (s p) ← ∼(w (s p))
w (s p) ← ∼(s p)
q (s q) ← ∼(w (s q))
w (s q) ← ∼(s q)
q (s w) ← ∼(w (s w))
w (s w) ← ∼(s w)
...
```

The well-founded model $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}$ of the above (infinite) propositional program is the valuation function of $\mathcal{M}^{WF}_{\mathsf{P}}$. It has already been argued in the discussion of Example 17, that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ p}) \neq \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ q})$ and this should be obvious to the reader who is familiar with the well-founded model semantics; however, for reasons of completeness, we present a formal argument in the second part of this proof. We also claimed that this is despite the fact that $\mathtt{p} \cong_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}},o\to o} \mathtt{q}$, which we will immediately proceed to prove. Of course, by Definitions 52 and 53, the facts that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ p}) \neq \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ q})$ and $\mathtt{p} \cong_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}},o\to o} \mathtt{q}$, render $\mathcal{M}^{WF}_{\mathsf{P}}$ not extensional.

First, we show that $\mathtt{p} \cong_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}},o\to o} \mathtt{q}$, i.e. that for all $\mathsf{A}, \mathsf{A}' \in U_{\mathsf{P},o}$ such that $\mathsf{A} \cong_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}},o} \mathsf{A}'$, $\mathtt{p\ A} \cong_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}},o} \mathtt{q\ A}'$ holds. By definition $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}$ is a fixed-point of the operator $\Theta_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}}}(\cdot)$, therefore for any ground atom B we have that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathsf{B})$ equals to T, if there exists a clause $\mathsf{B} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_n$ in $\mathsf{Gr(P)}$, such that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathsf{L}_i) = T$ for all $i \leq n$; it equals to F if for every clause $\mathsf{B} \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_n$ in $\mathsf{Gr(P)}$, we have that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathsf{L}_i) = F$ for at least one $i \leq n$; and it equals to 0 otherwise. Observe that there exists only one clause in $\mathsf{Gr(P)}$ such that $\mathtt{p\ A}$ is the head of the clause, in particular it is the ground instance $\mathtt{p\ A} \leftarrow \mathsf{A}$ of the clause $\mathtt{p\ R} \leftarrow \mathtt{R}$ of P. This suggests that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{p\ A}) = \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathsf{A})$ (1). Similarly, the ground instance $\mathtt{q\ A}' \leftarrow \sim(\mathtt{w\ A}')$ of the clause $\mathtt{q\ R} \leftarrow \sim(\mathtt{w\ R})$ is the only clause in $\mathsf{Gr(P)}$ with $\mathtt{q\ A}'$ as its head atom and from this we can infer that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{q\ A}') = \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\sim(\mathtt{w\ A}')) = \neg\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{w\ A}')$ (2). Finally, the only clause in $\mathsf{Gr(P)}$, such that $\mathtt{w\ A}'$ is the head of the clause, is the ground instance $\mathtt{w\ A}' \leftarrow \sim\mathsf{A}'$ of the clause $\mathtt{w\ R} \leftarrow \sim\mathtt{R}$ of P, which implies that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{w\ A}') = \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\sim\mathsf{A}') = \neg\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathsf{A}')$ (3). By (2) and (3) we have $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{q\ A}') = \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathsf{A}')$, and, in conjunction with (1), that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{p\ A}) = \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{q\ A}')$, because $\mathsf{A} \cong_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}},o} \mathsf{A}'$ implies, by Definition 52, that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathsf{A}) = \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathsf{A}')$. Therefore, we also have $\mathtt{p\ A} \cong_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}},o} \mathtt{q\ A}'$ and, in consequence, $\mathtt{p} \cong_{\mathsf{M}^{WF}_{\mathsf{Gr(P)}},o\to o} \mathtt{q}$.

For the second part, we show that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ p}) \neq \mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ q})$. We do this in two steps; first we show that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ p}) = F$ and then that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ q}) = 0$.

For the first step, it suffices to show that $\mathsf{M}_1(\mathtt{s\ p}) = \Theta^{\uparrow\omega}_{\mathsf{M}_0}(\mathtt{s\ p}) = F$. For this, we prove that $\Theta^{\uparrow n}_{\mathsf{M}_0}(\mathtt{s\ p}) = F$ and $\Theta^{\uparrow n}_{\mathsf{M}_0}(\mathtt{p\ (s\ p)}) = F$, for all $n < \omega$, by an induction on n. The basis case is trivial, as $\Theta^{\uparrow 0}_{\mathsf{M}_0} = \langle \emptyset, U_{\mathsf{P},o} \rangle$, assigns the value F to every atom. For the induction

step, we show the statement for $n+1$ assuming that it holds for n. We see that there exists only one clause in Gr(P) such that $\mathtt{s\ p}$ is the head of the clause; this is the ground instance $\mathtt{s\ p} \leftarrow \mathtt{p\ (s\ p)}$ of the clause $\mathtt{s\ Q} \leftarrow \mathtt{Q\ (s\ Q)}$ of P. By the induction hypothesis, we have that $\Theta^{\uparrow n}_{\mathsf{M}_0}(\mathtt{p\ (s\ p)}) = F$, therefore $\Theta^{\uparrow(n+1)}_{\mathsf{M}_0}(\mathtt{s\ p}) = F$. Similarly, the only clause in Gr(P) with $\mathtt{p\ (s\ p)}$ as the head of the clause is the ground instance $\mathtt{p\ (s\ p)} \leftarrow \mathtt{(s\ p)}$ of the clause $\mathtt{p\ R} \leftarrow \mathtt{R}$ of P. By the induction hypothesis, we have that $\Theta^{\uparrow n}_{\mathsf{M}_0}(\mathtt{s\ p}) = F$, therefore $\Theta^{\uparrow(n+1)}_{\mathsf{M}_0}(\mathtt{p\ (s\ p)}) = F$.

For the second step, we perform an induction on α, during which we simultaneously show that $\mathsf{M}_\alpha(\mathtt{s\ q}) = 0$, $\mathsf{M}_\alpha(\mathtt{q\ (s\ q)}) = 0$ and $\mathsf{M}_\alpha(\mathtt{w\ (s\ q)}) = 0$, for all countable ordinals α. The basis case is trivial, as $\mathsf{M}_0 = \langle \emptyset, \emptyset \rangle$ assigns the value 0 to all atoms. For the induction step, we first prove the statement for a successor ordinal $\alpha + 1$, assuming that it holds for all countable ordinals up to α. Indeed, there exists exactly one clause in Gr(P) with $\mathtt{w\ (s\ q)}$ as its head atom, in particular the ground instance $\mathtt{w\ (s\ q)} \leftarrow {\sim}\mathtt{(s\ q)}$ of the clause $\mathtt{q\ R} \leftarrow {\sim}\mathtt{(w\ R)}$. As ${\sim}\mathtt{(s\ q)}$ is a negative literal, for every $n < \omega$ the value of $\Theta^{\uparrow(n+1)}_{\mathsf{M}_\alpha}(\mathtt{w\ (s\ q)})$ is defined by $\mathsf{M}_\alpha({\sim}\mathtt{(s\ q)})$. By the induction hypothesis, we have $\mathsf{M}_\alpha(\mathtt{s\ q}) = \mathsf{M}_\alpha({\sim}\mathtt{(s\ q)}) = 0$, therefore it follows that $\Theta^{\uparrow(n+1)}_{\mathsf{M}_\alpha}(\mathtt{w\ (s\ q)}) = 0$ and, because this holds for every $n < \omega$, that $\mathsf{M}_{\alpha+1}(\mathtt{w\ (s\ q)}) = 0$. Moreover, the ground instance $\mathtt{q\ (s\ q)} \leftarrow {\sim}\mathtt{(w\ (s\ q))}$ of the clause $\mathtt{q\ R} \leftarrow {\sim}\mathtt{(w\ R)}$ is the only clause in Gr(P) with $\mathtt{q\ (s\ q)}$ as its head atom. Again, ${\sim}\mathtt{(w\ (s\ q))}$ is a negative literal and so for every $n < \omega$ the value of $\Theta^{\uparrow(n+1)}_{\mathsf{M}_\alpha}(\mathtt{q\ (s\ q)})$ only depends on $\mathsf{M}_\alpha({\sim}\mathtt{(w\ (s\ q))})$. By the induction hypothesis, we have $\mathsf{M}_\alpha(\mathtt{w\ (s\ q)}) = \mathsf{M}_\alpha({\sim}\mathtt{(w\ (s\ q))}) = 0$, therefore it follows that $\Theta^{\uparrow(n+1)}_{\mathsf{M}_\alpha}(\mathtt{q\ (s\ q)}) = 0$. Since this holds for every $n < \omega$, we also have that $\mathsf{M}_{\alpha+1}(\mathtt{q\ (s\ q)}) = 0$. Finally, there exists only one clause in Gr(P) such that $\mathtt{s\ q}$ is the head of the clause, in particular the ground instance $\mathtt{s\ q} \leftarrow \mathtt{q\ (s\ q)}$ of the clause $\mathtt{s\ Q} \leftarrow \mathtt{Q\ (s\ Q)}$ of P. We have already shown that $\Theta^{\uparrow(n+1)}_{\mathsf{M}_\alpha}(\mathtt{q\ (s\ q)}) = 0$ for all $n < \omega$; moreover, by the induction hypothesis, $\mathsf{M}_\alpha(\mathtt{q\ (s\ q)}) = 0$. Consequently, for all $n < \omega$, $\Theta^{\uparrow(n+2)}_{\mathsf{M}_\alpha}(\mathtt{s\ q}) = 0$ and thus $\mathsf{M}_{\alpha+1}(\mathtt{s\ q}) = 0$. It remains to show $\mathsf{M}_\alpha(\mathtt{s\ q}) = 0$, $\mathsf{M}_\alpha(\mathtt{q\ (s\ q)}) = 0$ and $\mathsf{M}_\alpha(\mathtt{w\ (s\ q)}) = 0$ for a limit ordinal α. In this case, we have that $\mathsf{M}_\alpha = \langle \bigcup_{\beta<\alpha} T_\beta, \bigcup_{\beta<\alpha} F_\beta \rangle$. By the induction hypothesis, $\mathsf{M}_\beta(\mathtt{s\ q}) = 0$ for all $\beta < \alpha$, which means that $\mathtt{s\ q} \notin T_\beta$ and $\mathtt{s\ q} \notin F_\beta$. In other words, $\mathtt{s\ q} \notin \bigcup_{\beta<\alpha} T_\beta$ and $\mathtt{s\ q} \notin \bigcup_{\beta<\alpha} F_\beta$, therefore $\mathsf{M}_\alpha(\mathtt{s\ q}) = 0$. In the same way we can show that $\mathsf{M}_\alpha(\mathtt{q\ (s\ q)}) = 0$ and $\mathsf{M}_\alpha(\mathtt{w\ (s\ q)}) = 0$. This concludes the induction and so we have proven that $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}(\mathtt{s\ q}) = 0$. □

Despite the above negative result, there exists a broad and useful class of programs that are extensional under the well-founded semantics. As we are going to see in Section 5.3.1, this class is the class of stratified programs.

5.2.3 Infinite-valued Extension

In this section we argue that the infinite-valued model adaptation of Bezem's technique is a more suitable candidate for capturing the extensional semantics of general $\mathcal{H}$ programs. The definition of the infinite-valued model of such a program is analogous to the definitions of stable models and the well-founded model of the previous sections:

Definition 65. Let P be a program. Also, let Gr(P) be the ground instantiation of P and let $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}$ be the infinite-valued model of Gr(P). We define $\mathcal{M}^{IV}_{\mathsf{P}}$ to be the infinite-valued Herbrand interpretation of P such that $\mathit{val}_{\mathcal{M}^{IV}_{\mathsf{P}}}(\mathsf{A}) = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A})$ for every $\mathsf{A} \in U_{\mathsf{P},o}$ and we call it the *infinite-valued model* of P.

Theorem 14 guarantees that $\mathcal{M}_{\mathsf{P}}^{IV}$ is the minimum Herbrand model of P, with respect to $\sqsubseteq$. However, unlike the stable and well-founded models of P, $\mathcal{M}_{\mathsf{P}}^{IV}$ *is* extensional.

Theorem 15. *The infinite-valued model $\mathcal{M}_{\mathsf{P}}^{IV}$ of every general program* P *is extensional.*

Proof. Since the valuation function of $\mathcal{M}_{\mathsf{P}}^{IV}$ is $\mathsf{M}_{\mathsf{Gr(P)}}^{WF}$, essentially we need to show that $\mathsf{E} \cong_{\mathsf{M}_{\mathsf{Gr(P)}}^{WF},\rho} \mathsf{E}$, for every ground expression E of every argument type ρ. We perform an induction on the structure of ρ. For the base types ι and o the statement holds by definition. For the induction step, we prove the statement for a predicate type $\pi = \rho_1 \to \cdots \to \rho_m \to o$, assuming that it holds for all types simpler than π (i.e., for the types $\rho_1, \ldots, \rho_m, o$ and, recursively, the types that are simpler than $\rho_1, \ldots, \rho_m$). Let A be any atom of the following form: A is headed by a predicate constant and all variables in $vars(\mathsf{A})$ are of types simpler than π. Let θ, θ' be ground substitutions, such that $vars(\mathsf{A}) \subseteq dom(\theta), dom(\theta')$ and $\theta(\mathsf{V}) \cong_{\mathsf{M}_{\mathsf{Gr(P)}}^{WF},\rho} \theta'(\mathsf{V})$ for any $\mathsf{V} : \rho$ in $vars(\mathsf{A})$. We claim it suffices to show the following two properties $P_1(\alpha)$ and $P_2(\alpha)$, for all ordinals α:

$P_1(\alpha)$**:** if $\mathsf{M}_\alpha(\mathsf{A}\theta) = T_\alpha$ then $\mathsf{M}_{\mathsf{Gr(P)}}^{WF}(\mathsf{A}\theta') = T_\alpha$;

$P_2(\alpha)$**:** if $\mathsf{M}_\alpha(\mathsf{A}\theta) = F_\alpha$ then $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{A}\theta') = F_\alpha$.

To see why proving the above properties is enough to establish that $\mathsf{E} \cong_{\mathsf{M}_{\mathsf{Gr(P)}}^{IV},\pi} \mathsf{E}$, observe the following: first of all, we assumed that π is of the form $\rho_1 \to \cdots \to \rho_m \to o$, so if $\mathsf{V}_1 : \rho_1, \ldots, \mathsf{V}_m : \rho_m$ are variables, then $\mathsf{E}\ \mathsf{V}_1\ \cdots\ \mathsf{V}_m$ is an atom of the form described above. Also, by Lemma 3 we have that $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{E}\ \theta(\mathsf{V}_1)\ \cdots\ \theta(\mathsf{V}_m)) = T_\alpha$ if and only if $\mathsf{M}_\alpha(\mathsf{E}\ \theta(\mathsf{V}_1)\ \cdots\ \theta(\mathsf{V}_m)) = T_\alpha$. If $P_1(\alpha)$ holds, this implies that $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{E}\ \theta'(\mathsf{V}_1)\ \cdots\ \theta'(\mathsf{V}_m)) = T_\alpha$. Because the relations $\cong_{\mathsf{M}_{\mathsf{Gr(P)}}^{IV},\rho_i}$ are symmetric, θ and θ' are interchangeable. Therefore, by the same argument, the reverse implication (i.e. $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{E}\ \theta'(\mathsf{V}_1)\ \cdots\ \theta'(\mathsf{V}_m)) = T_\alpha \Rightarrow \mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{E}\ \theta(\mathsf{V}_1)\ \cdots\ \theta(\mathsf{V}_m)) = T_\alpha$) can be inferred, thus resulting to an equivalence. If $P_2(\alpha)$ holds, the analogous equivalence can be shown for the value F_α, in the same way. Finally, the equivalence for the 0 value follows by a simple elimination argument: if for example $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{E}\ \theta(\mathsf{V}_1)\ \cdots\ \theta(\mathsf{V}_m)) = 0$, we make the assumption that $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{E}\ \theta'(\mathsf{V}_1)\ \cdots\ \theta'(\mathsf{V}_m)) = T_\alpha$ (respectively, F_α) for some ordinal α. Then, by Lemma 3, $\mathsf{M}_\alpha(\mathsf{E}\ \theta'(\mathsf{V}_1)\ \cdots\ \theta'(\mathsf{V}_m)) = T_\alpha$ (respectively, F_α), so if property $P_1(\alpha)$ (respectively, $P_2(\alpha)$) holds, it gives us that $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{E}\ \theta(\mathsf{V}_1)\ \cdots\ \theta(\mathsf{V}_m)) = T_\alpha$ (respectively, F_α), which is a contradiction. It follows that $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{E}\ \theta'(\mathsf{V}_1)\ \cdots\ \theta'(\mathsf{V}_m)) = 0$. Again, we can show the reverse implication by the same argument.

We will proceed by a second induction on α.

Second Induction Basis ($\alpha = 0$**)** We have $\mathsf{M}_0 = T_{\mathsf{P},0}^{\omega}(\emptyset)$. Observe that $T_{\mathsf{P},0}^{\omega}(\emptyset)(\mathsf{A}\theta)$ will evaluate to T_0 if and only if there exists some $n < \omega$ for which $T_{\mathsf{P}}^{n}(\emptyset)(\mathsf{A}\theta) = T_0$. On the other hand, it will evaluate to F_0 if and only if there does not exist a $n < \omega$ for which $T_{\mathsf{P}}^{n}(\emptyset)(\mathsf{A}\theta) \neq F_0$. Therefore, in order to prove $P_1(0)$ and $P_2(0)$, we first need to perform a third induction on n and prove the following two properties:

$P_1'(0,n)$**:** if $T_{\mathsf{P}}^{n}(\emptyset)(\mathsf{A}\theta) = T_0$ then $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{A}\theta') = T_0$;

$P_2'(0,n)$**:** if $T_{\mathsf{P}}^{n}(\emptyset)(\mathsf{A}\theta) > F_0$ then $\mathsf{M}_{\mathsf{Gr(P)}}^{IV}(\mathsf{A}\theta') > F_0$.

Third Induction Basis ($n = 0$) Both $P_1'(0,0)$ and $P_2'(0,0)$ hold vacuously, since $T_{\mathsf{P}}^0(\emptyset) = \emptyset$, i.e. the interpretation that assigns F_0 to every atom.

Third Induction Step ($n+1$) First we show $P_1'(0, n+1)$, assuming that $P_1'(0,n)$ holds. If $T_{\mathsf{P}}^{n+1}(\emptyset)(\mathsf{A}\theta) = T_0$, then there exists a clause $\mathsf{A}\theta \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_k$ in $\mathsf{Gr}(\mathsf{P})$ such that for each $i \leq k$, $T_{\mathsf{P}}^n(\emptyset)(\mathsf{L}_i) = T_0$. This implies that each L_i is a positive literal, since a negative one cannot be assigned the value T_0 in any interpretation. This clause is a ground instance of a clause $\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m \leftarrow \mathsf{B}_1, \ldots, \mathsf{B}_k$ in the higher-order program and there exists a substitution θ'', such that $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' = \mathsf{A}$ and, for any variable $\mathsf{V} \notin \{\mathsf{V}_1, \ldots, \mathsf{V}_m\}$ appearing in the body of the clause, $\theta''(\mathsf{V})$ is an appropriate ground term, so that $\mathsf{L}_i = \mathsf{B}_i\theta''\theta$ for all $i \leq k$. Observe that the variables appearing in the clause $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' \leftarrow \mathsf{B}_1\theta'', \ldots, \mathsf{B}_k\theta''$ are exactly the variables appearing in A and they are all of types simpler than π. We distinguish the following cases for each $\mathsf{B}_i\theta''$, $i \leq k$:

1. $\mathsf{B}_i\theta''$ *is of the form* $(\mathsf{E}_1 \approx \mathsf{E}_2)$*:* By definition, $T_{\mathsf{P}}^n(\emptyset)(\mathsf{L}_i) = T_{\mathsf{P}}^n(\emptyset)(\mathsf{B}_i\theta''\theta) = T_0$ implies that $\mathsf{E}_1\theta = \mathsf{E}_2\theta$. Since E_1 and E_2 are expressions of type ι, all variables in E_1 and E_2 are also of type ι and, because $\cong_{\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})},\iota}$ is defined as equality, we will have $\mathsf{E}_1\theta = \mathsf{E}_1\theta'$ and $\mathsf{E}_2\theta = \mathsf{E}_2\theta'$. Therefore $\mathsf{E}_1\theta' = \mathsf{E}_2\theta'$ and $\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\mathsf{B}_i\theta''\theta') = T_0$ will also hold.

2. $\mathsf{B}_i\theta''$ *is an atom and starts with a predicate constant:* As we observed, the variables appearing in $\mathsf{B}_i\theta''$ are of types simpler than π. By the third induction hypothesis, $\mathsf{B}_i\theta''$ satisfies property $P_1'(0,n)$ and therefore $T_{\mathsf{P}}^n(\emptyset)(\mathsf{L}_i) = T_{\mathsf{P}}^n(\emptyset)(\mathsf{B}_i\theta''\theta) = T_0$ implies that $\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\mathsf{B}_i\theta''\theta') = T_0$.

3. $\mathsf{B}_i\theta''$ *is an atom and starts with a predicate variable:* Let $\mathsf{B}_i\theta'' = \mathsf{V}\,\mathsf{E}_1 \cdots \mathsf{E}_{m'}$ for some $\mathsf{V} \in vars(\mathsf{A})$. Then $\mathsf{B} = \theta(\mathsf{V})\,\mathsf{E}_1 \cdots \mathsf{E}_{m'}$ is an atom that begins with a predicate constant and, by $vars(\mathsf{B}_i\theta'') \subseteq vars(\mathsf{A})$, all of the variables of B are of types simpler than π. Also, $T_{\mathsf{P}}^n(\emptyset)(\mathsf{B}\theta) = T_{\mathsf{P}}^n(\emptyset)(\mathsf{B}_i\theta''\theta) = T_0$, which, by property $P_1'(0,n)$ yields that $\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\mathsf{B}\theta') = \mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = T_0$ (1). Observe that the types of all arguments of $\theta(\mathsf{V})$, i.e. the types of $\mathsf{E}_j\theta'$ for all $j \leq m'$, are simpler than the type of V and consequently, since $\mathsf{V} \in vars(\mathsf{A})$, simpler than π. For each $j \leq m'$, let ρ_j be the type of E_j and let ρ be the type of V; by the first induction hypothesis, $\mathsf{E}_j\theta' \cong_{\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})},\rho_j} \mathsf{E}_j\theta'$. Moreover, by assumption we have that $\theta(\mathsf{V}) \cong_{\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})},\rho} \theta'(\mathsf{V})$. Then, by definition $\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = \mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')$ and, by (1), $\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = T_0$.

In conclusion, the clause $\mathsf{A}\theta' \leftarrow \mathsf{B}_1\theta''\theta', \ldots, \mathsf{B}_k\theta''\theta'$ is in $\mathsf{Gr}(\mathsf{P})$ and for each $i \leq k$, we have $\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\mathsf{B}_i\theta''\theta') = T_0$, therefore $\mathsf{M}^{IV}_{\mathsf{Gr}(\mathsf{P})}(\mathsf{A}\theta') = T_0$ must also hold.

This concludes the proof for $P_1'(0,n)$. Next we prove $P_2'(0,n+1)$, assuming $P_2'(0,n)$ holds. If $T_{\mathsf{P}}^{n+1}(\emptyset)(\mathsf{A}\theta) > F_0$, then there exists a clause $\mathsf{A}\theta \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_k$ in $\mathsf{Gr}(\mathsf{P})$ such that for each $i \leq k$, $T_{\mathsf{P}}^n(\emptyset)(\mathsf{L}_i) > F_0$. This clause is a ground instance of a clause $\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m \leftarrow \mathsf{K}_1, \ldots, \mathsf{K}_k$ in the P and there exists a substitution θ'', such that $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' = \mathsf{A}$ and, for any variable $\mathsf{V} \notin \{\mathsf{V}_1, \ldots, \mathsf{V}_m\}$ appearing in the body of the clause, $\theta''(\mathsf{V})$ is an appropriate ground term, so that $\mathsf{L}_i = \mathsf{K}_i\theta''\theta$ for all $i \leq k$. Observe that the variables appearing in the clause $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' \leftarrow \mathsf{K}_1\theta'', \ldots, \mathsf{K}_k\theta''$ are exactly the variables appearing in A and they are all of types simpler than π. We can distinguish the following cases for each $\mathsf{K}_i\theta''$, $i \leq k$:

1. $\mathsf{K}_i\theta''$ *is a positive literal:* A positive literal may take one of the three forms that we examined in our proof for property $P_1'(0, n+1)$. We can show that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') > F_0$ by the same arguments we used in each case.

2. $\mathsf{K}_i\theta''$ *is a negative literal:* A negative literal cannot be assigned the value F_0 in any interpretation. Therefore $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') > F_0$ holds by definition.

In conclusion, the clause $\mathsf{A}\theta' \leftarrow \mathsf{K}_1\theta''\theta', \ldots, \mathsf{K}_k\theta''\theta'$ is in $\mathsf{Gr(P)}$ and for each $i \leq k$, we have $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') > F_0$, therefore $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') > F_0$ must also hold.

This concludes the proof for $P_2'(0, n)$. We will now use properties $P_1'(0, n)$ and $P_2'(0, n)$ in order to show $P_1(0)$ and $P_2(0)$. By definition, if $\mathsf{M}_0(\mathsf{A}\theta) = T^{\omega}_{\mathsf{P},0}(\emptyset)(\mathsf{A}\theta) = T_0$, then there exists some $n < \omega$ such that $T^n_{\mathsf{P}}(\emptyset)(\mathsf{A}\theta) = T_0$. Applying $P_1'(0, n)$ to $\mathsf{A}\theta$ we immediately conclude that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = T_0$, which establishes property $P_1(0)$. Now let $\mathsf{M}_0(\mathsf{A}\theta) = F_0$ and assume $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') \neq F_0$. By Lemma 3, the latter can only hold if $\mathsf{M}_0(\mathsf{A}\theta') = T^{\omega}_{\mathsf{P},0}(\emptyset)(\mathsf{A}\theta') \neq F_0$ and this, in turn, means that there exists at least one $n < \omega$ such that $T^n_{\mathsf{P}}(\emptyset)(\mathsf{A}\theta') > F_0$. Then, reversing the roles of θ and θ', we can apply property $P_2'(0, n)$ to $\mathsf{A}\theta'$ and conclude that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta) > F_0$, which, again by Lemma 3, contradicts $\mathsf{M}_0(\mathsf{A}\theta) = F_0$. Therefore it must be $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = F_0$.

Second Induction Step Now we prove properties $P_1(\alpha)$ and $P_2(\alpha)$ for an arbitrary countable ordinal α, assuming that $P_1(\beta)$ and $P_2(\beta)$ hold for all $\beta < \alpha$.

We have $\mathsf{M}_\alpha = T^{\omega}_{\mathsf{P},\alpha}(\mathsf{I}_\alpha)$. Again, we first perform a third induction on n and prove two auxilary properties, as follows:

$P_1'(\alpha, n)$**:** if $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{A}\theta) \geq T_\alpha$ then $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') \geq T_\alpha$;

$P_2'(\alpha, n)$**:** if $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{A}\theta) > F_\alpha$ then $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') > F_\alpha$.

Third Induction Basis ($n = 0$**)** We have $T^0_{\mathsf{P}}(\mathsf{I}_\alpha) = \mathsf{I}_\alpha$. Observe that, whether α is a successor or a limit ordinal, I_α does not assign to any atom the value T_α or any value that is greater than F_α and smaller than T_α So if $T^0_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{A}\theta) \geq T_\alpha$ (respectively, $> F_\alpha$), it must be $T^0_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{A}\theta) = T_\beta$ for some ordinal $\beta < \alpha$. By Lemma 1, $\mathsf{M}_\alpha(\mathsf{A}\theta) = T_\beta$ and so, by Lemma 3, $\mathsf{M}_\beta(\mathsf{A}\theta) = T_\beta$. Then, by the second induction hypothesis, property $P_1(\beta)$ holds and yields $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = T_\beta \geq T_\alpha$ (respectively, $> F_\alpha$). Therefore property $P_1'(\alpha, 0)$ (respectively, $P_2'(\alpha, 0)$) also holds.

Third Induction Step ($n + 1$**)** First we show $P_1'(\alpha, n + 1)$, assuming that $P_1'(\alpha, n)$ holds. If $T^{n+1}_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{A}\theta) \geq T_\alpha$, then there exists a clause $\mathsf{A}\theta \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_k$ in $\mathsf{Gr(P)}$ such that for each $i \leq k$, $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{L}_i) \geq T_\alpha$. This clause is a ground instance of a clause $\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m \leftarrow \mathsf{K}_1, \ldots, \mathsf{K}_k$ in P and there exists a substitution θ'', such that $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' = \mathsf{A}$ and, for any variable $\mathsf{V} \notin \{\mathsf{V}_1, \ldots, \mathsf{V}_m\}$ appearing in the body of the clause, $\theta''(\mathsf{V})$ is an appropriate ground term, so that $\mathsf{L}_i = \mathsf{K}_i\theta''\theta$ for all $i \leq k$. As we observed earlier, the variables appearing in the clause $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' \leftarrow \mathsf{K}_1\theta'', \ldots, \mathsf{K}_k\theta''$ are exactly the variables appearing in A and they are all of types simpler than π. We can distinguish the following cases for each $\mathsf{K}_i\theta''$:

1. $\mathsf{K}_i\theta''$ *is a positive literal:* A positive literal may take one of the three forms that we examined in our proof for property $P_1'(0, n+1)$. We can show that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') \geq T_\alpha$ by the same arguments we used in each case.

2. $\mathsf{K}_i\theta''$ *is a negative literal and its atom starts with a predicate constant:* Let $\mathsf{K}_i\theta''$ be of the form $\sim\mathsf{B}$, where B is an atom that starts with a predicate constant. It is $T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\sim\mathsf{B}\theta) = T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{K}_i\theta''\theta) = T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{L}_i) \geq T_\alpha$. Then $T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{B}\theta) < F_\alpha$, i.e. $T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{B}\theta) = F_\beta$ for some ordinal $\beta < \alpha$. By Lemma 1, $\mathsf{M}_\alpha(\mathsf{B}\theta) = F_\beta$ and thus, by Lemma 3, $\mathsf{M}_\beta(\mathsf{B}\theta) = F_\beta$. By $vars(\mathsf{K}_i\theta'') \subseteq vars(\mathsf{A})$, all the variables of B are of types simpler than π, so we can apply the second induction hypothesis, in particular property $P_2(\beta)$, to $\mathsf{B}\theta$ and conclude that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') = F_\beta$. Then $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim\mathsf{B}\theta') = T_{\beta+1} \geq T_\alpha$.

3. $\mathsf{K}_i\theta''$ *is a negative literal and its atom starts with a predicate variable:* Let $\mathsf{K}_i\theta'' = \sim(\mathsf{V}\,\mathsf{E}_1 \cdots \mathsf{E}_{m'})$ for some $\mathsf{V} \in vars(\mathsf{A})$. Then $\mathsf{B} = \theta(\mathsf{V})\,\mathsf{E}_1 \cdots \mathsf{E}_{m'}$ is an atom that begins with a predicate constant and, by $vars(\mathsf{K}_i\theta'') \subseteq vars(\mathsf{A})$, all the variables of B are of types simpler than π. Also, $T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\sim\mathsf{B}\theta) = T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{K}_i\theta''\theta) = T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{L}_i) \geq T_\alpha$. Then $T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{B}\theta) < F_\alpha$, i.e. $T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{B}\theta) = F_\beta$ for some ordinal $\beta < \alpha$. By Lemma 1, $\mathsf{M}_\alpha(\mathsf{B}\theta) = F_\beta$ and thus, by Lemma 3, $\mathsf{M}_\beta(\mathsf{B}\theta) = F_\beta$. By the second induction hypothesis, property $P_2(\beta)$ gives us that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = F_\beta$ (1). The types of all arguments of $\theta(\mathsf{V})$, i.e. the types of $\mathsf{E}_j\theta'$ for all $j \leq m'$, are simpler than the type of V and consequently, since $\mathsf{V} \in vars(\mathsf{A})$, simpler than π. For each $j \leq m'$, let ρ_j be the type of E_j and let ρ be the type of V; by the first induction hypothesis, $\mathsf{E}_j\theta' \cong_{\mathsf{M}^{IV}_{\mathsf{Gr(P)}},\rho_j} \mathsf{E}_j\theta'$. Moreover, by assumption we have that $\theta(\mathsf{V}) \cong_{\mathsf{M}^{IV}_{\mathsf{Gr(P)}},\rho} \theta'(\mathsf{V})$. Then, by definition $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')$ and, by (1), $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = F_\beta$. Therefore, it follows that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')) = T_{\beta+1} \geq T_\alpha$.

We can conclude that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') \geq T_\alpha$, as the clause $\mathsf{A}\theta' \leftarrow \mathsf{K}_1\theta''\theta', \ldots, \mathsf{K}_k\theta''\theta'$ is in $\mathsf{Gr(P)}$ and we have shown that, for each $i \leq k$, $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') \geq T_\alpha$.

This concludes the proof of $P_1'(\alpha, n)$. Next we prove $P_2'(\alpha, n+1)$, assuming $P_2'(\alpha, n)$ holds. If $T_{\mathsf{P}}^{n+1}(\mathsf{I}_\alpha)(\mathsf{A}\theta) > F_\alpha$, then there exists a clause $\mathsf{A}\theta \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_k$ in $\mathsf{Gr(P)}$ such that for each $i \leq k$, $T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{L}_i) > F_\alpha$. This clause is a ground instance of a clause $\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m \leftarrow \mathsf{K}_1, \ldots, \mathsf{K}_k$ in P and there exists a substitution θ''such that $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' = \mathsf{A}$ and, for any variable $\mathsf{V} \notin \{\mathsf{V}_1, \ldots, \mathsf{V}_m\}$ appearing in the body of the clause, $\theta''(\mathsf{V})$ is an appropriate ground term, so that $\mathsf{L}_i = \mathsf{K}_i\theta''\theta$ for all $i \leq k$. Again we observe that the variables appearing in the clause $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' \leftarrow \mathsf{K}_1\theta'', \ldots, \mathsf{K}_k\theta''$ are exactly the variables appearing in A, which are all of types simpler than π, and distinguish the following cases for each $\mathsf{K}_i\theta''$:

1. $\mathsf{K}_i\theta''$ *is a positive literal:* A positive literal may take one of the three forms that we examined in our proof for property $P_1'(0, n+1)$. We can show that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') > F_\alpha$ by the same arguments we used in each case.

2. $\mathsf{K}_i\theta''$ *is a negative literal and its atom starts with a predicate constant:* Let $\mathsf{K}_i\theta''$ be of the form $\sim\mathsf{B}$, where B is an atom that starts with a predicate constant and, by $vars(\mathsf{K}_i\theta'') \subseteq vars(\mathsf{A})$, all the variables of B are of types simpler than π. It is $T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\sim\mathsf{B}\theta) = T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{K}_i\theta''\theta) = T_{\mathsf{P}}^n(\mathsf{I}_\alpha)(\mathsf{L}_i) > F_\alpha$ and we claim that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim\mathsf{B}\theta') > F_\alpha$. For the sake of contradiction, assume that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim\mathsf{B}\theta') \leq F_\alpha$. Then $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') > T_\alpha$, i.e. $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') = T_\beta$ for some ordinal $\beta < \alpha$. Therefore, Lemma 3 implies that $\mathsf{M}_\beta(\mathsf{B}\theta') = T_\beta$. This means that, if we reverse the roles of θ and θ', we can apply the second induction hypothesis to $\mathsf{B}\theta'$ and use property $P_1(\beta)$ to conclude that

$\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{B}\theta) = T_\beta$. By Lemma 3, this implies that $\mathsf{M}_\alpha(\mathsf{B}\theta) = T_\beta$ and, by Lemma 1, $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{B}\theta) = T_\beta$. Then $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\sim\mathsf{B}\theta) = F_{\beta+1} \leq F_\alpha$, which is obviously a contradiction. So it must be $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim\mathsf{B}\theta') > F_\alpha$.

3. $\mathsf{K}_i\theta''$ *is a negative literal and its atom starts with a predicate variable:* Let $\mathsf{K}_i\theta'' = \sim(\mathsf{V}\,\mathsf{E}_1 \cdots \mathsf{E}_{m'})$ for some $\mathsf{V} \in vars(\mathsf{A})$. Then $\mathsf{B} = \theta(\mathsf{V})\,\mathsf{E}_1 \cdots \mathsf{E}_{m'}$ is an atom that begins with a predicate constant and, by $vars(\mathsf{K}_i\theta'') \subseteq vars(\mathsf{A})$, all the variables of B are of types simpler than π. Also, $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\sim\mathsf{B}\theta) = T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{K}_i\theta''\theta) = T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{L}_i) > F_\alpha$. We claim that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim\mathsf{B}\theta') > F_\alpha$. Again, assume that this is not so, i.e. $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim\mathsf{B}\theta') \leq F_\alpha$; then $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') = T_\beta > T_\alpha$ for some ordinal $\beta < \alpha$. By Lemma 3, $\mathsf{M}_\beta(\mathsf{B}\theta') = T_\beta$ and the second induction hypothesis applies. So if we reverse the roles of θ and θ' property $P_1(\beta)$ gives us that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{B}\theta) = T_\beta$. By Lemma 3, $\mathsf{M}_\alpha(\mathsf{B}\theta) = T_\beta$ and thus, by Lemma 1, $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{B}\theta) = T_\beta$. This is a contradiction, as it implies that $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\sim\mathsf{B}\theta) = F_{\beta+1} \leq F_\alpha$. Hence it must hold that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim\mathsf{B}\theta') = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')) > F_\alpha$ (1). The types of all arguments of $\theta(\mathsf{V})$, i.e. the types of $\mathsf{E}_j\theta'$ for all $j \leq m'$, are simpler than the type of V and therefore simpler than π. For each $j \leq m'$, let ρ_j be the type of E_j and let ρ be the type of V; by the first induction hypothesis, $\mathsf{E}_j\theta' \cong_{\mathsf{M}^{IV}_{\mathsf{Gr(P)}},\rho_j} \mathsf{E}_j\theta'$ and by assumption $\theta(\mathsf{V}) \cong_{\mathsf{M}^{IV}_{\mathsf{Gr(P)}},\rho} \theta'(\mathsf{V})$. Then, by definition $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')$ and, consequently, $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')) = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta'))$. Therefore by (1), $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')) > F_\alpha$ and so we can conclude that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = \mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\sim(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')) > F_\alpha$.

Observe that the clause $\mathsf{A}\theta' \leftarrow \mathsf{K}_1\theta''\theta', \ldots, \mathsf{K}_k\theta''\theta'$ is in $\mathsf{Gr(P)}$ and we have shown that, for each $i \leq k$, $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') > F_\alpha$. So $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') > F_\alpha$ must also hold.

This concludes the proof for $P'_2(\alpha, n)$. We will now use properties $P'_1(\alpha, n)$ and $P'_2(\alpha, n)$ in order to show $P_1(\alpha)$ and $P_2(\alpha)$. By definition, if $\mathsf{M}_\alpha(\mathsf{A}\theta) = T^\omega_{\mathsf{P},\alpha}(\mathsf{I}_\alpha)(\mathsf{A}\theta) = T_\alpha$, then there exists some $n < \omega$ such that $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{A}\theta) = T_\alpha$. As we have shown above, property $P'_1(\alpha, n)$ yields $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') \geq T_\alpha$. However, if it was $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = T_\beta > T_\alpha$ for some ordinal $\beta < \alpha$, then, by Lemma 3 we would also have $\mathsf{M}_\beta(\mathsf{A}\theta') = T_\beta$. By the second induction hypothesis we would be able to apply property $P_1(\beta)$ to $\mathsf{A}\theta'$ and infer that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta) = T_\beta$, which, again by Lemma 3, contradicts $\mathsf{M}_\alpha(\mathsf{A}\theta) = T_\alpha$. So $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta')$ can only be equal to T_α and property $P_1(\alpha)$ holds. Now let $\mathsf{M}_\alpha(\mathsf{A}\theta) = F_\alpha$ and assume $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') \neq F_\alpha$, i.e. either $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') < F_\alpha$ or $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') > F_\alpha$. In the first case, $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = F_\beta$ and, by Lemma 3, $\mathsf{M}_\beta(\mathsf{A}\theta') = F_\beta$ for some $\beta < \alpha$. By the second induction hypothesis, property $P_2(\beta)$ gives us that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta) = F_\beta < F_\alpha$. In the second case, Lemma 3 implies that $\mathsf{M}_\alpha(\mathsf{A}\theta') = T^\omega_{\mathsf{P},\alpha}(\mathsf{I}_\alpha)(\mathsf{A}\theta') > F_\alpha$ and this, in turn, means that there exists at least one $n < \omega$ such that $T^n_{\mathsf{P}}(\mathsf{I}_\alpha)(\mathsf{A}\theta') > F_\alpha$. Then, reversing the roles of θ and θ', we can apply property $P'_2(\alpha, n)$ to $\mathsf{A}\theta'$ and conclude that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta) > F_\alpha$. In both cases, our conclusion constitutes a contradiction, because, by Lemma 3, $\mathsf{M}_\alpha(\mathsf{A}\theta) = F_\alpha$ implies that $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta) = F_\alpha$. Therefore it must also be $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = F_\alpha$ and this proves property $P_2(\alpha)$. □

5.3 Stratified and Locally Stratified Programs

In this section we define the notions of *stratified* and *locally stratified* higher-order logic programs and show that the class of stratified programs is well-behaved with respect to

extensionality. The notion of local stratification is a straightforward generalization of the corresponding notion for classical (first-order) logic programs. However, the notion of stratification is a genuine extension of the corresponding notion for first-order programs.

Definition 66. A program P is called *locally stratified* if and only if it is possible to decompose the Herbrand base $U_{\mathsf{P},o}$ of P into disjoint sets (called *strata*) $S_1, S_2, \ldots, S_\alpha, \ldots, \alpha < \gamma$, where γ is a countable ordinal, such that for every clause $\mathsf{H} \leftarrow \mathsf{A}_1, \ldots, \mathsf{A}_m, \sim\mathsf{B}_1, \ldots, \sim\mathsf{B}_n$ in Gr(P), we have that for every $i \leq m$, *stratum*$(\mathsf{A}_i) \leq$ *stratum*(H) and for every $i \leq n$, *stratum*$(\mathsf{B}_i) <$ *stratum*(H), where *stratum* is a function such that *stratum*$(\mathsf{C}) = \beta$, if the atom $\mathsf{C} \in U_{\mathsf{P},o}$ belongs to S_β, and *stratum*$(\mathsf{C}) = 0$, if $\mathsf{C} \notin U_{\mathsf{P},o}$ and is of the form $(\mathsf{E}_1 \approx \mathsf{E}_2)$.

Since Definition 66 generalizes the corresponding one for classical logic programs, the undecidability result [12] for detecting whether a given program is locally stratified, extends directly to the higher-order case.

Lemma 12. *The problem of determining whether a given logic program* P *is locally stratified, is undecidable.*

However, there exists a notion of stratification for higher-order logic programs that is decidable and has as a special case the stratification for classical logic programs [1]. In the following definition, a predicate type π is understood to be *greater than* a second predicate type π', if π is of the form $\rho_1 \to \cdots \to \rho_n \to \pi'$, where $n \geq 1$.

Definition 67. A program P is called *stratified* if and only if it is possible to decompose the set of all predicate constants that appear in P into a finite number r of disjoint sets (called *strata*) $S_1, S_2, \ldots, S_r$, such that for every clause $\mathsf{H} \leftarrow \mathsf{A}_1, \ldots, \mathsf{A}_m, \sim\mathsf{B}_1, \ldots, \sim\mathsf{B}_n$ in P, where the predicate constant of H is p, we have:

1. for every $i \leq m$, if A_i is a term starting with a predicate constant q, then *stratum*$(\mathsf{q}) \leq$ *stratum*(p);
2. for every $i \leq m$, if A_i is a term starting with a predicate variable Q, then for all predicate constants q that appear in P such that the type of q is greater than or equal to the type of Q, it holds *stratum*$(\mathsf{q}) \leq$ *stratum*(p);
3. for every $i \leq n$, if B_i starts with a predicate constant q, then *stratum*$(\mathsf{q}) <$ *stratum*(p);
4. for every $i \leq n$, if B_i starts with a predicate variable Q, then for all predicate constants q that appear in P such that the type of q is greater than or equal to the type of Q, it holds *stratum*$(\mathsf{q}) <$ *stratum*(p);

where *stratum*$(\mathsf{r}) = i$ if the predicate constant r belongs to S_i.

Example 18. In the following program:

```
p Q ← ∼(Q a)
q X ← (X≈a)
```

the variable `Q` is of type $\iota \to o$ and `X` is of type ι. The only predicate constants that appear in the program are `p`, which is of type $(\iota \to o) \to o$, and `q`, which is of type $\iota \to o$. Note that the type of `p` is neither equal nor greater than the type of `Q`, while the type of `q` is the same as that of `Q`. It is straightforward to see that the program is stratified, if we choose $S_1 = \{\mathtt{q}\}$

and $S_2 = \{\mathtt{p}\}$. Indeed, for the first clause, we have $stratum(\mathtt{p}) > stratum(\mathtt{q})$ and in the second clause there are no predicate constants or predicate variables appearing in its body. However, if `Q` and `X` are as above and, moreover, `Y` is of type ι, it can easily be checked that the program:

```
p Q ← ∼(Q a)
q X Y ← (X≈a), (Y≈a), p (q a)
```

is not stratified nor locally stratified, because if the term `q a` is substituted for `Q` we get a circularity through negation. Notice that the type of `q` is $\iota \to \iota \to o$ and it is greater than the type of `Q` which is $\iota \to o$. □

Since the set of predicate constants that appear in a program P is finite, and since the number of predicate constants of the program that have a greater or equal type than the type of a given predicate variable is also finite, it follows that checking whether a given program is stratified, is decidable. Moreover, we have the following theorem:

Theorem 16. *If* P *is stratified then it is locally stratified.*

Proof. Consider a decomposition $S_1, \ldots, S_r$ of the set of predicate constants of P such that the requirements of Definition 67 are satisfied. This defines a decomposition $S'_1, \ldots, S'_r$ of the Herbrand base of P, as follows:

$$S'_i = \{\mathsf{A} \in U_{\mathsf{P},o} \mid \text{the leftmost predicate constant of A belongs to } S_i\}$$

We show that $S'_1, \ldots, S'_r$ corresponds to a local stratification of $U_{\mathsf{P},o}$. Consider a clause in P of the form $\mathsf{H}' \leftarrow \mathsf{A}'_1, \ldots, \mathsf{A}'_m, \sim\mathsf{B}'_1, \ldots, \sim\mathsf{B}'_n$ and let $\mathsf{H} \leftarrow \mathsf{A}_1, \ldots, \mathsf{A}_m, \sim\mathsf{B}_1, \ldots, \sim\mathsf{B}_n$ be one of its ground instances. Let p be the predicate constant of H (and H′). Consider any A'_i. If A'_i starts with a predicate constant, say q_i, by Definition 67, it is $stratum(\mathsf{p}) \geq stratum(\mathsf{q}_i)$. By the definition of the local stratification decomposition we gave above, it is $stratum(\mathsf{H}) = stratum(\mathsf{p})$ and $stratum(\mathsf{A}_i) = stratum(\mathsf{q}_i)$, and therefore $stratum(\mathsf{H}) \geq stratum(\mathsf{A}_i)$. If A'_i starts with a predicate variable, say Q, then Q has been substituted in A'_i with a term starting with a predicate constant, say q_i, that has a type greater than or equal to that of Q. By Definition 67, it is $stratum(\mathsf{p}) \geq stratum(\mathsf{q}_i)$ and by the definition of the local stratification decomposition we gave above, it is $stratum(\mathsf{H}) = stratum(\mathsf{p})$ and $stratum(\mathsf{A}_i) = stratum(\mathsf{q}_i)$, and therefore $stratum(\mathsf{H}) \geq stratum(\mathsf{A}_i)$. The justification for the case of negative literals, is similar and omitted. □

5.3.1 Semantics of Stratified Programs

In this section we focus on stratified higher-order programs and show that this class of programs has some nice properties under the extensions of Bezem's semantics, which we defined in Section 5.2.

First of all, as in the first-order case, all atoms in the infinite-valued and well-founded model of a locally stratified program have non-zero values. Recall that, by Theorem 16, stratified programs are a subset of locally stratified ones, therefore the lemma also holds for stratified programs.

Lemma 13. *Let* P *be a locally stratified logic program. Then, for every atom* $\mathsf{A} \in U_{\mathsf{P},o}$ *it holds* $val_{\mathcal{M}_\mathsf{P}^{WF}}(\mathsf{A}) \neq 0$ *and* $val_{\mathcal{M}_\mathsf{P}^{IV}}(\mathsf{A}) \neq 0$.

Proof. Theorem 5 implies that the infinite-valued model $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}$ of the ground instantiation of P assigns the truth value 0 to an atom iff this atom is assigned the truth value 0 by the well-founded model of $\mathsf{Gr(P)}$. Notice now that, by Definition 66, if P is a locally stratified higher-order program, then $\mathsf{Gr(P)}$ is in turn a locally stratified propositional program (having exactly the same local stratification as P). Recall that the well-founded model of a locally stratified propositional program does not assign the truth value 0 to any atom [18], so neither does $\mathcal{M}^{WF}_{\mathsf{P}}$, or $\mathsf{M}^{IV}_{\mathsf{Gr(P)}}$ or, consequently, $\mathcal{M}^{IV}_{\mathsf{P}}$. □

However, the most interesting property of stratified programs is the fact that their well-founded models are always extensional, even though this is not true in the general case (see Lemma 11). This is demonstrated in Theorem 17 and the next lemma is the basis for our proof of this theorem:

Lemma 14. *Let* P *be an* $\mathcal{H}$ *program. If* P *is stratified then it has a unique stable model, which coincides with its well-founded model and with the collapse of its infinite-valued model.*

Proof. Let $\mathsf{Gr(P)}$ be the ground instantiation of P. By Theorem 16, $\mathsf{Gr(P)}$ is locally stratified. Therefore, by Theorem 2, $\mathsf{Gr(P)}$ has a unique stable model which coincides with the perfect model of $\mathsf{Gr(P)}$. As a consequence, P also has a unique stable model. By Theorem 5, the interpretation obtained by collapsing all true values of the infinite-valued model of $\mathsf{Gr(P)}$ to T and all false values to F, coincides with the well-founded model of $\mathsf{Gr(P)}$. Again, because $\mathsf{Gr(P)}$ is locally stratified, by Theorem 3, the well-founded model of $\mathsf{Gr(P)}$ coincides with its perfect model. In conclusion, the unique stable model, the well-founded model and the collapse of the infinite-valued model of P are all two-valued and coincide with each other, because the valuation function of each of these models coincides with the perfect model of $\mathsf{Gr(P)}$. □

Theorem 17. *The well-founded model* $\mathcal{M}^{WF}_{\mathsf{P}}$ *of a stratified program* P *is extensional.*

Proof. By Theorem 16, if P is stratified then $\mathsf{Gr(P)}$ is locally stratified. Therefore the unique perfect model $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}$ of $\mathsf{Gr(P)}$ exists (Theorem 1). Moreover, the perfect model of a locally stratified propositional program coincides with its well-founded model $\mathsf{M}^{WF}_{\mathsf{Gr(P)}}$ (Theorem 3), i.e. the valuation function of $\mathcal{M}^{WF}_{\mathsf{P}}$. So, to show that $\mathcal{M}^{WF}_{\mathsf{P}}$ is extensional in the case of stratified programs, we can rely upon the constructive definition of $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}$ from [28] presented in Section 3.4.

Consider a stratification $S_1, \ldots, S_r$ of the set of predicate constants of P. As argued in the proof of Theorem 16, the following decomposition $S'_1, \ldots, S'_r$ of $U_{\mathsf{P},o}$:

$$S'_i = \{\mathsf{A} \in U_{\mathsf{P},o} \mid \text{the leftmost predicate constant of A belongs to } S_i\}$$

corresponds to a local stratification of $\mathsf{Gr(P)}$. So, whenever a ground atom A begins with a predicate constant p, we will have $stratum(\mathsf{A}) = stratum(\mathsf{p})$. Moreover, by Theorem 1, $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}} = \mathsf{N}_r$.

Since the valuation function of $\mathcal{M}^{WF}_{\mathsf{P}}$ is $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}$, essentially we need to show that $\mathsf{E} \cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\rho} \mathsf{E}$, for every ground expression E of every argument type ρ. We perform an induction on the structure of ρ. For the base types ι and o the statement holds by definition. For the induction step, we prove the statement for a predicate type $\pi = \rho_1 \to \cdots \to \rho_m \to o$, assuming that it holds for all types simpler than π (i.e., for the types $\rho_1, \ldots, \rho_m, o$ and,

recursively, the types that are simpler than $\rho_1, \ldots, \rho_m$). Let A be any atom of the following form: A is headed by a predicate constant p and all variables in $vars(\mathsf{A})$ are of types simpler than π. Let θ, θ' be ground substitutions, such that $vars(\mathsf{A}) \subseteq dom(\theta), dom(\theta')$ and $\theta(\mathsf{V}) \cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\rho} \theta'(\mathsf{V})$ for any $\mathsf{V} : \rho$ in $vars(\mathsf{A})$. We claim it suffices to show the following two properties $P_1(\alpha)$ and $P_2(\alpha)$, for all finite ordinals (i.e., natural numbers) α:

$P_1(\alpha)$**:** If $\mathsf{N}_\alpha(\mathsf{A}\theta) = T$ then $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = T$.

$P_2(\alpha)$**:** If $\mathsf{N}_\alpha(\mathsf{A}\theta) = F$ then $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = F$.

To see why proving the above properties is enough to establish that $\mathsf{E} \cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\pi} \mathsf{E}$, observe the following: first of all, we assumed that π is of the form $\rho_1 \to \cdots \to \rho_m \to o$, so if $\mathsf{V}_1 : \rho_1, \ldots, \mathsf{V}_m : \rho_m$ are variables, then $\mathsf{E}\,\mathsf{V}_1 \cdots \mathsf{V}_m$ is an atom of the form described above. As $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}} = \mathsf{N}_r$, if $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{E}\,\theta(\mathsf{V}_1) \cdots \theta(\mathsf{V}_m)) = \mathsf{N}_r(\mathsf{E}\,\theta(\mathsf{V}_1) \cdots \theta(\mathsf{V}_m)) = T$ and property $P_1(r)$ holds, then we can infer that $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{E}\,\theta'(\mathsf{V}_1) \cdots \theta'(\mathsf{V}_m)) = T$. Because the relations $\cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\rho_i}$ are symmetric, θ and θ' are interchangeable. Therefore the same argument can be used to infer the reverse implication, i.e. $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{E}\,\theta'(\mathsf{V}_1) \cdots \theta'(\mathsf{V}_m)) = T \Rightarrow \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{E}\,\theta(\mathsf{V}_1) \cdots \theta(\mathsf{V}_m)) = T$, thus resulting to an equivalence. If $P_2(r)$ holds, the analogous equivalence can be shown for the value F in the same way and so it follows that $\mathsf{E} \cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\pi} \mathsf{E}$. Finally, r is determined by the stratification of the higher-order program and is therefore finite, so we only need to prove properties $P_1(\alpha)$ and $P_2(\alpha)$ for finite ordinals.

We will proceed by a second induction on α.

Second Induction Basis ($\alpha = 0$**)** We have $\mathsf{N}_0 = \langle \emptyset, \emptyset \rangle$. As this interpretation does not assign the value T or the value F to any atom, both properties $P_1(0)$ and $P_2(0)$ hold vacuously.

Second Induction Step ($\alpha{+}1$**)** We first show $P_1(\alpha{+}1)$. We have that $\mathsf{N}_{\alpha+1} = \langle \Psi^{\uparrow\omega}_{\mathsf{N}_\alpha}, \mathcal{B}_{\alpha+1} - \Psi^{\uparrow\omega}_{\mathsf{N}_\alpha} \rangle$; observe that $\Psi^{\uparrow\omega}_{\mathsf{N}_\alpha}(\mathsf{A}\theta) = T$ if and only if there exists some $n < \omega$ for which $\Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{A}\theta) = T$. Therefore, in order to prove $P_1(\alpha + 1)$, we first need to perform a third induction on n and prove the following property:

$P'_1(\alpha + 1, n)$**:** If $\Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{A}\theta) = T$ then $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = T$.

Third Induction Basis ($n = 0$**)** Property $P'_1(\alpha + 1, 0)$ holds vacuously, since $\Psi^{\uparrow 0}_{\mathsf{N}_\alpha} = \emptyset$, i.e. it does not assign the value T to any atom.

Third Induction Step ($n + 1$**)** We now show property $P'_1(\alpha + 1, n + 1)$, assuming that $P'_1(\alpha+1, n)$ holds. If $\Psi^{\uparrow(n+1)}_{\mathsf{N}_\alpha}(\mathsf{A}\theta) = T$, then there exists a clause $\mathsf{A}\theta \leftarrow \mathsf{L}_1, \ldots, \mathsf{L}_k$ in $\mathsf{Gr(P)}$ such that, for each $i \leq k$, either $\mathsf{N}_\alpha(\mathsf{L}_i) = T$ or L_i is an atom and $\Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{L}_i) = T$. This clause is a ground instance of a clause $\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m \leftarrow \mathsf{K}_1, \ldots, \mathsf{K}_k$ in the higher-order program and there exists a substitution θ'', such that $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' = \mathsf{A}$ and, for any variable $\mathsf{V} \notin \{\mathsf{V}_1, \ldots, \mathsf{V}_m\}$ appearing in the body of the clause, $\theta''(\mathsf{V})$ is an appropriate ground term, so that $\mathsf{L}_i = \mathsf{K}_i\theta''\theta$ for all $i \leq k$. Observe that the variables appearing in the clause $(\mathsf{p}\,\mathsf{V}_1 \cdots \mathsf{V}_m)\theta'' \leftarrow \mathsf{K}_1\theta'', \ldots, \mathsf{K}_k\theta''$ are exactly the variables appearing in A and they are all of types simpler than π. We distinguish the following cases for each $\mathsf{K}_i\theta''$, $i \leq k$:

1. $\mathsf{K}_i\theta''$ *is of the form* $(\mathsf{E}_1 \approx \mathsf{E}_2)$*:* An expression of the form $(\mathsf{E}_1 \approx \mathsf{E}_2)$ has the same value in any interpretation. If $\mathsf{N}_\alpha(\mathsf{K}_i\theta''\theta) = \Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{K}_i\theta''\theta) = T$, by definition we have $\mathsf{E}_1\theta = \mathsf{E}_2\theta$. Since E_1 and E_2 are expressions of type ι, all variables in E_1 and E_2 are also of type ι and, because $\cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\iota}$ is defined as equality, we will have $\mathsf{E}_1\theta = \mathsf{E}_1\theta'$ and $\mathsf{E}_2\theta = \mathsf{E}_2\theta'$. Therefore $\mathsf{E}_1\theta' = \mathsf{E}_2\theta'$ and $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = T$ will also hold.

2. $\mathsf{K}_i\theta''$ *is an atom and starts with a predicate constant:* As we observed, the variables appearing in $\mathsf{K}_i\theta''$ are of types simpler than π. Because $\mathsf{K}_i\theta''\theta$ is an atom, either $\mathsf{N}_\alpha(\mathsf{L}_i) = \mathsf{N}_\alpha(\mathsf{K}_i\theta''\theta) = T$ or $\Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{L}_i) = \Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{K}_i\theta''\theta) = T$ may hold. In the former case, by the second induction hypothesis we can apply property $P_1(\alpha)$ and it follows that $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = T$. Similarly, in the latter case, the same conclusion can be reached by the third induction hypothesis and property $P'_1(\alpha+1, n)$.

3. $\mathsf{K}_i\theta''$ *is an atom and starts with a predicate variable:* As in the previous case, it may be $\mathsf{N}_\alpha(\mathsf{L}_i) = \mathsf{N}_\alpha(\mathsf{K}_i\theta''\theta) = T$ or $\Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{L}_i) = \Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{K}_i\theta''\theta) = T$. Let $\mathsf{K}_i\theta'' = \mathsf{V}\,\mathsf{E}_1 \cdots \mathsf{E}_{m'}$ for some $\mathsf{V} \in vars(\mathsf{A})$. Then $\mathsf{B} = \theta(\mathsf{V})\,\mathsf{E}_1 \cdots \mathsf{E}_{m'}$ is an atom that begins with a predicate constant and, by $vars(\mathsf{K}_i\theta'') \subseteq vars(\mathsf{A})$, all of the variables of B are of types simpler than π. Hence, by the second induction hypothesis, B satisfies property $P_1(\alpha)$ and if $\mathsf{N}_\alpha(\mathsf{K}_i\theta''\theta) = \mathsf{N}_\alpha(\mathsf{B}\theta) = T$ then it follows that $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') = T$ (1). Similarly, by the third induction hypothesis, B also satisfies property $P'_1(\alpha+1, n)$, so if $\Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{K}_i\theta''\theta) = \Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{B}\theta) = T$, then the same conclusion, that $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') = T$ (1), is reached again. Observe that the types of all arguments of $\theta(\mathsf{V})$, i.e. the types of $\mathsf{E}_j\theta'$ for all $j \leq m'$, are simpler than the type of V and consequently, since $\mathsf{V} \in vars(\mathsf{A})$, simpler than π. For each $j \leq m'$, let ρ_j be the type of E_j and let ρ be the type of V; by the first induction hypothesis, $\mathsf{E}_j\theta' \cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\rho_j} \mathsf{E}_j\theta'$. Moreover, by assumption we have that $\theta(\mathsf{V}) \cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\rho} \theta'(\mathsf{V})$. Then, by definition and by (1) $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = T$.

4. $\mathsf{K}_i\theta''$ *is a negative literal and its atom starts with a predicate constant:* Let $\mathsf{K}_i\theta''$ be of the form $\sim\mathsf{B}$, where B is an atom that starts with a predicate constant. It is $\mathsf{N}_\alpha(\sim\mathsf{B}\theta) = \mathsf{N}_\alpha(\mathsf{K}_i\theta''\theta) = \mathsf{N}_\alpha(\mathsf{L}_i) = T$ and therefore $\mathsf{N}_\alpha(\mathsf{B}\theta) = F$. Moreover, by $vars(\mathsf{K}_i\theta'') \subseteq vars(\mathsf{A})$, all the variables of B are of types simpler than π, so we can apply the second induction hypothesis, in particular property $P_2(\alpha)$, to B and conclude that $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') = F$. Then $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\sim\mathsf{B}\theta') = \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = T$.

5. $\mathsf{K}_i\theta''$ *is a negative literal and its atom starts with a predicate variable:* Let $\mathsf{K}_i\theta'' = \sim(\mathsf{V}\,\mathsf{E}_1 \cdots \mathsf{E}_{m'})$ for some $\mathsf{V} \in vars(\mathsf{A})$. Then $\mathsf{B} = \theta(\mathsf{V})\,\mathsf{E}_1 \cdots \mathsf{E}_{m'}$ is an atom that begins with a predicate constant and, by $vars(\mathsf{K}_i\theta'') \subseteq vars(\mathsf{A})$, all of the variables of B are of types simpler than π. Also, $\mathsf{N}_\alpha(\sim\mathsf{B}\theta) = \mathsf{N}_\alpha(\mathsf{K}_i\theta''\theta) = \mathsf{N}_\alpha(\mathsf{L}_i) = T$ and therefore $\mathsf{N}_\alpha(\mathsf{B}\theta) = F$. Hence, by the second induction hypothesis and in particular property $P_2(\alpha)$, it follows that $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{B}\theta') = \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = F$ (1). Observe that the types of all arguments of $\theta(\mathsf{V})$, i.e. the types of $\mathsf{E}_j\theta'$ for all $j \leq m'$, are simpler than the type of V and consequently, since $\mathsf{V} \in vars(\mathsf{A})$, simpler than π. For each $j \leq m'$, let ρ_j be the type of E_j and let ρ be the type of V; by the first induction hypothesis, $\mathsf{E}_j\theta' \cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\rho_j} \mathsf{E}_j\theta'$. Moreover, by assumption we have that $\theta(\mathsf{V}) \cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}},\rho} \theta'(\mathsf{V})$. Then, by definition and by (1), $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\theta(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta') = F$. So, obviously, $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\sim(\theta'(\mathsf{V})\,\mathsf{E}_1\theta' \cdots \mathsf{E}_{m'}\theta')) = \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = T$.

We have shown that, for each $i \leq k$, $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{K}_i\theta''\theta') = T$. Since the clause $\mathsf{A}\theta' \leftarrow \mathsf{K}_1\theta''\theta', \ldots, \mathsf{K}_k\theta''\theta'$ is in $\mathsf{Gr}(\mathsf{P})$ and $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}$ is a model of $\mathsf{Gr}(\mathsf{P})$, we conclude that $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = T$.

This concludes the proof for $P'_1(\alpha+1, n)$. Notice that property $P'_1(\alpha+1, n)$ immediately implies property $P_1(\alpha+1)$: as mentioned before, $\mathsf{N}_{\alpha+1}(\mathsf{A}\theta) = \Psi^{\uparrow\omega}_{\mathsf{N}_\alpha}(\mathsf{A}\theta) = T$ if and only if there exists some $n < \omega$ for which $\Psi^{\uparrow n}_{\mathsf{N}_\alpha}(\mathsf{A}\theta) = T$ and then $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = T$ follows from property $P'_1(\alpha+1, n)$. It remains to prove property $P_2(\alpha+1)$. Observe that the atoms $\mathsf{A}\theta$ and $\mathsf{A}\theta'$ both start with the same predicate constant p and recall that we have chosen a local stratification for $\mathsf{Gr}(\mathsf{P})$, such that $stratum(\mathsf{A}\theta) = stratum(\mathsf{A}\theta') = stratum(\mathsf{p})$. Moreover, we have that $\mathsf{N}_{\alpha+1} = \langle \Psi^{\uparrow\omega}_{\mathsf{N}_\alpha}, \mathcal{B}_{\alpha+1} - \Psi^{\uparrow\omega}_{\mathsf{N}_\alpha} \rangle$, so if $\mathsf{N}_{\alpha+1}(\mathsf{A}\theta) = F$, it follows that $\mathsf{A}\theta \in \mathcal{B}_{\alpha+1}$. Because $stratum(\mathsf{A}\theta) = stratum(\mathsf{A}\theta')$, it must also be $\mathsf{A}\theta' \in \mathcal{B}_{\alpha+1}$, which implies that $\mathsf{N}_{\alpha+1}(\mathsf{A}\theta')$ can be either T (if $\mathsf{A}\theta' \in \Psi^{\uparrow\omega}_{\mathsf{N}_\alpha}$) or F (if $\mathsf{A}\theta' \notin \Psi^{\uparrow\omega}_{\mathsf{N}_\alpha}$), but not 0. For the sake of contradiction, assume that $\mathsf{N}_{\alpha+1}(\mathsf{A}\theta') = T$. As the relations $\cong_{\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}, \rho_i}$ are symmetric, θ and θ' are interchangeable, so property $P_1(\alpha+1)$ applies and yields $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{A}\theta) = T$. Because (by Theorem 1) $\mathsf{N}_{\alpha+1} \preceq \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}$, this contradicts our initial assumption that $\mathsf{N}_{\alpha+1}(\mathsf{A}\theta) = F$. Therefore, it must be $\mathsf{N}_{\alpha+1}(\mathsf{A}\theta') = F$ and so, again by $\mathsf{N}_{\alpha+1} \preceq \mathsf{M}^{Pe}_{\mathsf{Gr(P)}}$, it follows that $\mathsf{M}^{Pe}_{\mathsf{Gr(P)}}(\mathsf{A}\theta') = F$. □

An immediate consequence of the above theorem, is that the unique stable model and the collapse of the infinite valued model of a stratified program are also extensional, since they coincide with its well-founded model by Lemma 14.

6. CONCLUSIONS AND FUTURE WORK

We have shown that the two existing extensional approaches to positive higher-order logic programming ([31, 9] and [2, 3]) coincide for the programs of the language considered in [31], but differ in general when existential predicate variables are allowed in clause bodies. We have presented an additional condition under which the two approaches continue to coincide even for this more general class of programs.

Moreover, we have for the first time adapted Bezem's technique of [2, 3] to define an extensional semantics for higher-order programs with negation. For this purpose, we have utilized the infinite-valued approach to negation-as-failure [30]. On the other hand, we have shown that an adaptation of the technique under the well-founded or the stable model semantics does not in general lead to an extensional semantics. Finally, we have defined the notions of stratification and local stratification and proven that the class of stratified programs is a notable exception to the previous negative result.

Despite the fact that stratified programs lead to an extensional well-founded model, we have not been able to verify that the same property holds for *locally stratified higher-order logic programs*. On the other hand, our attempts to find a locally stratified program with a non-extensional well-founded model have also been unsuccessful, and therefore it is not at present clear to us whether this class of programs is well-behaved with respect to extensionality, or not.

Another matter worth looking into is the relationships between the extensions of Bezem's semantics presented in Chapter 5 and their domain theoretic counterparts. For example, we would expect that the infinite-valued semantics that we have developed and the domain theoretic infinite-valued semantics of [8] have close connections similar to those shared by the respective semantics for positive programs. A more intriguing question, however, arises from considering the three-valued extensions of Bezem's semantics and the domain theoretic semantics. It is natural to wonder if the failure of Bezem's approach under the well-founded (and the stable model) semantics is an inherent shortcoming of the technique or a more general problem.

We re-examine the counterexample of Section 5.2.2 but now with this broader question in mind. In particular, we indicate that in order to achieve an extensional three-valued semantics for higher-order logic programs with negation, one has to make some (arguably) non-standard assumptions regarding the behaviour of negation in such programs. In this respect, a logic with an infinite number of truth values appears to be a more appropriate vehicle for achieving extensionality. In the following discussion, we assume some basic familiarity with the main intuition behind the approach described in [8].

Consider again the program of Section 5.2.2. Under the infinite-valued adaptation of Bezem's approach given in Section 5.2.3 and also under the domain-theoretic infinite-valued approach of [8], the semantics of that program *is* extensional. The reason is that both of these approaches differentiate the meaning of p from the meaning of q. The truth domain in both approaches is the set $\mathbb{V}^{\infty}$, where F_{α} and T_{α} represent different degrees of truth and falsity. Under this truth domain, predicate p (intuitively) corresponds to the infinite-valued relation:

$$p = \{(v, v) \mid v \in \mathbb{V}^{\infty}\}$$

while predicate q corresponds to the relation:

$$q = \{(F_{\alpha}, F_{\alpha+2}) \mid \alpha < \Omega\} \cup \{(0, 0)\} \cup \{(T_{\alpha}, T_{\alpha+2}) \mid \alpha < \Omega\}$$

where Ω is the first uncountable ordinal. Obviously, the relations p and q are different as sets and therefore it is not a surprise that under both the semantics of Section 5.2.3 and of [8], the atoms `(s p)` and `(s q)` have different truth values. Notice, however, that if we collapse p and q in three-valued logic (i.e., if we map each F_α to F, each T_α to T, and 0 to 0), the collapsed relations become identical.

Assume now that we want to devise an (alternative to the one presented in Section 5.2.2) extensional three-valued semantics for $\mathcal{H}$ programs. Under such a semantics, it seems reasonable to assume that `p` and `q` would correspond to the same three-valued relation, namely $\{(v, v) \mid v \in \{F, 0, T\}\}$. Notice however that from a logic programming perspective, `p` and `q` are expected to have a different operational behaviour when they appear inside a program. In particular, given the program:

```
s Q ← Q (s Q)
p R ← R
```

we expect the atom `(s p)` to have the value F (due to the circularity that occurs if we try to evaluate it), while given the program:

```
s Q ← Q (s Q)
q R ← ∼(w R)
w R ← ∼R
```

we expect the atom `(s q)` to have the value 0 due to the circularity through negation. At first sight, the above discussion seems to suggest that there is no way we can have a three-valued extensional semantics for all higher-order logic programs with negation.

However, the above discussion is based mainly on our experience regarding the behaviour of first-order logic programs with negation. One could argue that we could devise a semantics under which `(s q)` will also return the value F. One possible justification for such a semantics would be that the definition of `q` uses two negations which cancel each other, and therefore we should actually expect `q` to behave exactly like `p` when it appears inside a program. It is worth noting that such cancellations of double negations appear in certain semantic approaches to negation. For example, for certain extended propositional programs, the semantics based on approximation fixpoint theory has the effect of canceling double negations (see for example [15][page 185, Example 1]). This approach is used as the basis of the three-valued extensional semantics developed in [10].

We believe that a comparative evaluation of the merits of the approach of [10] and the infinite-valued approaches of Section 5.2.3 and of [8] is an interesting project for future research.

REFERENCES

[1] Krzysztof R. Apt, Howard A. Blair, and Adrian Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, 1988.

[2] Marc Bezem. Extensionality of simply typed logic programs. In Danny De Schreye, editor, *Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999*, pages 395–410. MIT Press, 1999.

[3] Marc Bezem. An improved extensionality criterion for higher-order logic programs. In Laurent Fribourg, editor, *Computer Science Logic, 15th International Workshop, CSL 2001. 10th Annual Conference of the EACSL, Paris, France, September 10-13, 2001, Proceedings*, volume 2142 of *Lecture Notes in Computer Science*, pages 203–216. Springer, 2001.

[4] Marc Bezem. Hoapata programs are monotonic. In *Proceedings NWPT'02, Institute of Cybernetics at TTU, Tallinn*, pages 18–20, 2002.

[5] Stephen L. Bloom and Zoltán Ésik. *Iteration Theories - The Equational Logic of Iterative Processes*. EATCS Monographs on Theoretical Computer Science. Springer, 1993.

[6] Arnaud Carayol and Zoltán Ésik. An analysis of the equational properties of the well-founded fixed point. In Chitta Baral, James P. Delgrande, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016.*, pages 533–536. AAAI Press, 2016.

[7] Angelos Charalambidis. *Proof Procedures for Extensional Higher-Order Logic Programming*. PhD thesis, Athens, Greece, 2014.

[8] Angelos Charalambidis, Zoltán Ésik, and Panos Rondogiannis. Minimum model semantics for extensional higher-order logic programming with negation. *TPLP*, 14(4-5):725–737, 2014.

[9] Angelos Charalambidis, Konstantinos Handjopoulos, Panos Rondogiannis, and William W. Wadge. Extensional higher-order logic programming. *ACM Trans. Comput. Log.*, 14(3):21, 2013.

[10] Angelos Charalambidis, Panos Rondogiannis, and Ioanna Symeonidou. Approximation fixpoint theory and the well-founded semantics of higher-order logic programs (*in press*). 2018.

[11] Weidong Chen, Michael Kifer, and David Scott Warren. Hilog: A foundation for higher-order logic programming. *The Journal of Logic Programming*, 15(3):187 – 230, 1993.

[12] Peter Cholak and Howard A. Blair. The complexity of local stratification. *Fundam. Inform.*, 21(4):333–344, 1994.

[13] Jan Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[14] Keith L. Clark. Negation as failure. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, 1977.*, Advances in Data Base Theory, pages 293–322, New York, 1977. Plemum Press.

[15] Marc Denecker, Maurice Bruynooghe, and Joost Vennekens. Approximation fixpoint theory and the semantics of logic and answers set programs. In Esra Erdem, Joohyung Lee, Yuliya Lierler, and David Pearce, editors, *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz*, volume 7265 of *Lecture Notes in Computer Science*, pages 178–194. Springer, 2012.

[16] Zoltán Ésik. Equational properties of stratified least fixed points (extended abstract). In Valeria de Paiva, Ruy J. G. B. de Queiroz, Lawrence S. Moss, Daniel Leivant, and Anjolina Grisi de Oliveira, editors, *Logic, Language, Information, and Computation - 22nd International Workshop, WoLLIC 2015, Bloomington, IN, USA, July 20-23, 2015, Proceedings*, volume 9160 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 2015.

[17] Allen Van Gelder. Negation as failure using tight derivations for general logic programs. *J. Log. Program.*, 6(1&2):109–133, 1989.

[18] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.

[19] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, August 15-19, 1988 (2 Volumes)*, pages 1070–1080. MIT Press, 1988.

[20] John Hughes. Why functional programming matters. *Comput. J.*, 32:98–107, 1989.

[21] Vassilis Kountouriotis, Panos Rondogiannis, and William W. Wadge. Extensional higher-order datalog. In *Short Paper Proceedings of the 12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, pages 1–5, December 2005.

[22] Vladimir Lifschitz. Twelve definitions of a stable model. In Maria Garcia de la Banda and Enrico Pontelli, editors, *Logic Programming*, pages 37–51, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
[23] John W. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987.
[24] Dale Miller and Gopalan Nadathur. Higher-order logic programming. In *Proceedings of the Third International Conference on Logic Programming (ICLP)*, pages 448–462, 1986.
[25] Gopalan Nadathur. *A Higher-order Logic As the Basis for Logic Programming*. PhD thesis, Philadelphia, PA, USA, 1987. UMI Order No. GAX87-14099.
[26] Gopalan Nadathur and Dale Miller. Higher-order logic programming. In *Handbook of Logics for Artificial Intelligence and Logic Programming*, volume 5, pages 499–590. Clarendon Press, 1998.
[27] Halina Przymusinska and Teodor C. Przymusinski. Weakly perfect model semantics for logic programs. In *ICLP/SLP*, 1988.
[28] Halina Przymusinska and Teodor C. Przymusinski. Semantic issues in deductive databases and logic programs. In *Formal Techniques in Artificial Intelligence*, pages 321–367. North-Holland, 1990.
[29] Teodor C. Przymusinski. Foundations of deductive databases and logic programming. chapter On the Declarative Semantics of Deductive Databases and Logic Programs, pages 193–216. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
[30] Panos Rondogiannis and William W. Wadge. Minimum model semantics for logic programs with negation-as-failure. *ACM Trans. Comput. Log.*, 6(2):441–467, 2005.
[31] William W. Wadge. Higher-order horn logic programming. In Vijay A. Saraswat and Kazunori Ueda, editors, *Logic Programming, Proceedings of the 1991 International Symposium, San Diego, California, USA, Oct. 28 - Nov 1, 1991*, pages 289–303. MIT Press, 1991.
[32] David H. D. Warren. *Higher-order extensions to Prolog - are they needed*. Department of Artificial Intelligence, University of Edinburgh, 1981.