Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Εφαρμοσμενων Μαθηματικων και Φυσικων Επιστημων
Τομεας Μαθηματικων

Έλεγχος τύπων διεργασιών του π-λογισμού ως προς πολιτικές ιδιωτικότητας και μια εκτελέσιμη υλοποίησή του στην Maude

Διπλωματική εργασία

Γεώργιος Β. Πιτσιλαδής

Επιβλέπων
Πέτρος Στεφανέας

14 Νοεμβρίου 2016

Type checking privacy policies in the π-calculus and an executable implementation in Maude

Έλεγχος τύπων διεργασιών του π-λογισμού ως προς πολιτικές ιδιωτικότητας και μια εκτελέσιμη υλοποίησή του στην Maude

g.v.pitsiladis@gmail.com

Περίληψη

Η παρούσα εργασία παρουσιάζει ένα τυπικό σύστημα που επεκτείνει το πλαίσιο που έχει οριστεί από τους Kouzapas και συνεργάτες. Αποτελείται από μία γλώσσα έκφρασης πολιτικών ιδιωτικότητας, μία παραλλαγή του υπολογιστικού μοντέλου του π-λογισμού, καθώς και ένα σύστημα ελέγχου τύπων, το οποίο μπορεί να ελέγξει στατικά αν ένα σύστημα του π-λογισμού σέβεται μια δεδομένη πολιτική ιδιωτικότητας· η επέκταση έγκειται στην υποστήριξη προϋποθέσεων από γλώσσα έκφρασης πολιτικών ιδιωτικότητας και τα συστήματα του π-λογισμού. Αρχικά, ορίζονται πλήρως το συντακτικό και η σημασιολογία των μερών συστήματος και παρουσιάζονται μαθηματικές αποδείξεις της ορθότητάς του. Έπειτα, τα στοιχεία που απαιτούνται για τον έλεγχο τύπων υλοποιούνται σε εκτελέσιμη μορφή στη γλώσσα προγραμματισμού/προδιαγραφών Maude.

Λέξεις κλειδιά: ιδιωτικότητα, π-λογισμός, συστήματα τύπων, έλεγχος πρόσβασης, Maude

Abstract

This thesis presents a formal system which builds upon the privacy framework defined by Kouzapas et al. The formal system is comprised by a privacy policy language, a variance of the computational model of π-calculus and a type checking system, which can statically infer whether a π-calculus system respects a privacy policy; the privacy policy language and π-calculus are extended to support conditions. Initially, the syntax and semantics of the parts of the system are thoroughly discussed and proofs of its correctness are carried out. Then, the crucial parts for type checking are implemented as executable specifications in the Maude specification/programming language.

Keywords: privacy, π-calculus, type systems, access control, Maude

Ευχαριστίες

Κατ' αρχάς, θα ήθελα να ευχαριστήσω τους γονείς μου, που μου έδωσαν τα υλικά και πνευματικά εφόδια με τα οποία ακόμη πορεύομαι. Ένα τεράστιο «ευχαριστώ» ανήκει δικαιωματικά στη σύντροφό μου, Στεύη Θεοδοσίου, επειδή υπέμεινε το αυξανόμενο άγχος μου όσο καιρό ασχολήθηκα με την εργασία και με στήριξε ποικιλλοτρόπως.

Ευχαριστώ τον επιβλέποντα καθηγητή μου, Πέτρο Στεφανέα, για τις κατευθύνσεις και τη βοήθεια που μου έδωσε –τόσο ως προς την ίδια την εργασία όσο και ως προς την «αξιοποίησή» της–, καθώς και τα μέλη της εξεταστικής επιτροπής, Αριστείδη Αραγεώργη και Αλέξανδρο Αρβανιτάκη.

Για τη συγγραφή του κειμένου χρησιμοποιήθηκε η γλώσσα στοιχειοθεσίας LaTeX. Δεν θα μπορούσα να τη χρησιμοποιήσω αν δεν είχαν δουλέψει πάνω της δεκάδες άνθρωποι μέχρι σήμερα, είτε γράφοντας μακροεντολές είτε αναρτώντας τις εμπειρίες τους με αυτήν στο Διαδίκτυο.

Περιεχόμενα

Περίληψη **ii**

Abstract **iii**

Εισαγωγή **vii**

1 Η φύση και η ανάγκη προστασίας της ιδιωτικότητας **1**
1.1 Γιατί είναι αναγκαία η προστασία της ιδιωτικότητας 1
1.2 Ο ρόλος των νέων τεχνολογιών 4
1.3 Ορισμοί και προσεγγίσεις της έννοιας της ιδιωτικότητας . . . 5
1.3.1 Η πληροφοριακή οντολογία του Floridi 5
1.3.2 Η ιδιωτικότητα ως δικαίωμα και η θεωρία των φραγμών του Rickless 6
1.3.3 Η ταξινομία του Solove 7
1.3.4 Ο τέλειος ηδονοβλεψίας 9
1.4 Προσπάθειες προστασίας της ιδιωτικότητας 10
1.4.1 Το νομικό πλαίσιο στην ΕΕ και την Ελλάδα 10
1.4.2 Ιδιωτικότητα εκ σχεδιασμού 10

2 Έλεγχος τύπων, η Maude και τα θεμέλιά της **12**
2.1 Τι είναι και πού στοχεύει ο έλεγχος τύπων 12
2.2 Τα μαθηματικά θεμέλια της Maude 13
2.2.1 Συστήματα αναγωγών 13
2.2.2 Η εξισωτική λογική 14
2.2.3 Η λογική της αναγραφής 15
2.3 Η Maude 15
2.3.1 Βασικά συντακτικά και σημασιολογικά στοιχεία 16
2.3.2 Δηλώσεις 16
2.3.3 Ισχυρισμοί 18
2.3.4 Ιεραρχίες αρθρωμάτων 20
2.3.5 Περαιτέρω δυνατότητες της Maude 20

3 Έλεγχος τύπων προγραμμάτων του π-λογισμού ως προς πολιτικές ιδιωτικότητας **21**
3.1 Η γλώσσα περιγραφής των πολιτικών ιδιωτικότητας 21
3.1.1 Βασικοί ορισμοί 22
3.1.2 Μερικές χρήσιμες σχέσεις διάταξης 24
3.1.3 Ένα παράδειγμα πολιτικής ιδιωτικότητας 27

3.2 Ο π-λογισμός 30
3.2.1 Συντακτικό 31
3.2.2 Σημασιολογία 33
3.2.3 Παραδείγματα συστημάτων του π-λογισμού 37
3.3 Το σύστημα ελέγχου τύπων 40
3.3.1 Ορισμοί για τον έλεγχο τύπων 40
3.3.2 Συμβατότητα συστήματος με πολιτική 43
3.3.3 Παραδείγματα χρήσης του συστήματος ελέγχου τύπων 45
3.4 Αποδείξεις ορθότητας 47

4 Η εκτελέσιμη υλοποίηση του ελέγχου τύπων σε Maude 58
4.1 Η γλώσσα περιγραφής των πολιτικών ιδιωτικότητας 59
4.2 Ο π-λογισμός 64
4.3 Το σύστημα ελέγχου τύπων 67
4.4 Παραδείγματα χρήσης της υλοποίησης 74

Α΄ Ο πλήρης κώδικας της υλοποίησης 78

Βιβλιογραφία 91

Εισαγωγή

Η ιδιωτικότητα είναι ένα δικαίωμα του οποίου η προστασία έχει αρχίσει να αποκτάει όλο και μεγαλύτερη σημασία. Αν και η ακριβής φύση της δεν έχει καθοριστεί, έχει σχετιστεί με την ανθρώπινη αξιοπρέπεια, με την ατομική αυτονομία, καθώς και με την διατήρηση διαφορετικών κοινωνικών σχέσεων· επιπλέον, έχουν παρατηρηθεί –και ταξινομηθεί– αρκετές διαφορετικές περιπτώσεις παραβίασής της. Δεδομένου ότι η τεχνολογία που μπορεί να οδηγήσει σε παραβίαση της ιδιωτικότητας αναπτύσσεται ραγδαία, το αντίστοιχο πρέπει να συμβεί και με την τεχνολογία που την προστατεύει· σε αυτό το πλαίσιο, υπάρχει μεγάλη ανάγκη για εργαλεία που είναι σχεδιασμένα για να σέβονται την ιδιωτικότητα των χρηστών τους και εργαλεία που να αποτρέπουν ή να εντοπίζουν πιθανές ή πραγματικές παραβιάσεις της.

Ένα είδος τέτοιων εργαλείων θα μπορούσαν να είναι κάποια ορθά, αποδοτικά και εύχρηστα τυπικά συστήματα που να μπορούν να χρησιμοποιηθούν για στατικό ή δυναμικό συλλογισμό πάνω σε ιδιότητες πληροφοριακών συστημάτων και για έλεγχο και εφαρμογή προδιαγραφών. Απλές εκδοχές τέτοιου είδους συστημάτων έχουν αρχίσει να εμφανίζονται στη βιβλιογραφία: στο [16] ορίζεται ένα πλαίσιο που χρησιμοποιεί έλεγχο τύπων για να ελέγξει ιδιότητες σχετικές με την ιδιωτικότητα σε εφαρμογές του διαδικτύου, ειδικά σε δεδομένα που χρησιμοποιούν πρότυπα όπως το Rich Document Format (RDF)· το [28] ορίζει ένα εκφραστικό τυπικό σύστημα βασισμένο στην επιστημική λογική, σχεδιασμένο για την περιγραφή πολιτικών ιδιωτικότητας σε κοινωνικά δίκτυα· το [19] –το πλαίσιο του οποίου επεκτείνεται σε αυτή την εργασία– ορίζει μία γλώσσα έκφρασης πολιτικών ιδιωτικότητας και ένα σύστημα ελέγχου τύπων που διαπιστώνει αν διεργασίες του π-λογισμού σέβονται πολιτικές ιδιωτικότητας.

Παράλληλα, δεδομένης της (ασθενούς) συσχέτισης μεταξύ ιδιωτικότητας και ασφάλειας, και συνακόλουθα του ότι οι πολιτικές ιδιωτικότητας έχουν κοινά στοιχεία με τις πολιτικές ασφάλειας, έχουν γίνει προσπάθειες να επεκταθούν αποτελέσματα του δεύτερου τομέα στον πρώτο· μία τέτοια προσπάθεια είναι το P-RBAC (Privacy-aware Role Based Access Control) [25, 26, 3], από το οποίο προέρχονται οι έννοιες που χρησιμοποιούνται εδώ για την επέκταση του πλαισίου του [19].

Εκτός από την προαναφερθείσα επέκταση, στην παρούσα εργασία παρουσιάζεται η εκτελέσιμη υλοποίηση του επεκτεταμένου πλαισίου στη γλώσσα Maude. Η επιλογή της Maude έγινε λόγω των στέρεων μαθηματικών θεμελίων της, του χαρακτήρα της που κινείται ανάμεσα σε γλώσσα προγραμματισμού και γλώσσα προδιαγραφών, τη δυνατότητά της να λειτουργεί και για την απόδειξη ιδιοτήτων και την διατεινόμενη αποδοτικότητά της [22].

Η εργασία είναι οργανωμένη σε τέσσερα κεφάλαια: στο πρώτο κεφάλαιο, γίνεται μια συζήτηση περί της έννοιας της ιδιωτικότητας και της προστασίας της· στο δεύτερο κεφάλαιο, αναφέρονται κάποια βασικά στοιχεία για την Maude, κυρίως εκείνα που θα χρειαστούν παρακάτω· στο τρίτο κεφάλαιο, παρουσιάζεται το τυπικό σύστημα που αποτελεί τον πυρήνα της εργασίας και αποτελείται από μία γλώσσα έκφρασης πολιτικών ιδιωτικότητας, μία παραλλαγή του υπολογιστικού μοντέλου του π-λογισμού και ένα σύστημα ελέγχου τύπων που διαπιστώνει αν ένα πρόγραμμα του π-λογισμού σέβεται μια πολιτική ιδιωτικότητας· στο τέταρτο κεφάλαιο, παρουσιάζεται αναλυτικά ο κώδικας της υλοποίησης του συστήματος ελέγχου τύπων στην Maude.

Το μεγαλύτερο κομμάτι του κεφαλαίου 3 και μία συνοπτική περίληψη του κεφαλαίου 4 έχουν παρουσιαστεί στην ημερίδα Formal Methods for Privacy, το Νοέμβριο του 2016 στην Κύπρο.

Κεφάλαιο 1

Η φύση και η ανάγκη προστασίας της ιδιωτικότητας

Παρόλο που –ειδικά τα τελευταία χρόνια– γίνεται (δικαιολογημένα) πολύς λόγος και πολλή προσπάθεια για την προστασία της ιδιωτικότητας, είναι γενική παραδοχή ότι δεν υπάρχει ένας κοινώς αποδεκτός ορισμός για αυτή ή μία κοινώς αποδεκτή αιτιολόγηση της ανάγκης προστασίας της. Μελετητές από διάφορες επιστήμες έχουν ασχοληθεί με αυτή την έννοια· εδώ θα κάνουμε μία συζήτηση πάνω στο τι είναι η ιδιωτικότητα, γιατί πρέπει να προστατεύεται, καθώς και πάνω σε κάποιες προσπάθειες προστασίας της, χρησιμοποιώντας ενδεικτικά λίγες από την πληθώρα των διαθέσιμων προσεγγίσεων από το χώρο της φιλοσοφίας, της νομικής επιστήμης, των οικονομικών και της κοινωνιολογίας.

Δεδομένης της απουσίας συνεκτικού θεωρητικού πλαισίου, θα προχωρήσουμε πρώτα σε μία άνευ στέρεης θεωρητικής βάσης υπεράσπιση της ιδιωτικότητας και έπειτα σε μερικές προσπάθειες καθορισμού της. Θα υπάρξουν επίσης ένας σύντομος σχολιασμός για το ρόλο των νέων τεχνολογιών, καθώς και για κάποιες προσπάθειες προστασίας της.

1.1 Γιατί είναι αναγκαία η προστασία της ιδιωτικότητας

Σε πολλές περιπτώσεις, οι πληροφορίες που θεωρούνται ιδιωτικές μπορούν –αν αποκαλυφθούν– να οδηγήσουν σε προσβολή της φήμης ή της προσωπικότητας του ατόμου που αφορούν· μερικές φορές, είναι δυνατόν να καταλήξουν και σε οικονομική ζημιά, όταν για παράδειγμα αφορούν ζητήματα που πρέπει να μείνουν κρυφά από αντίδικους, εργοδότες ή ασφαλιστικές εταιρείες.

Εξάλλου, η εν γνώσει του υποκειμένου παραβίαση της ιδιωτικότητάς του δημιουργεί αρνητικά συναισθήματα· μπορεί μάλιστα να έχει επιπτώσεις στη συμπεριφορά του, ακόμη περισσότερο όταν συμβαίνει δυνητικά οποτεδή-

ποτε. Την τελευταία παρατήρηση έκανε ο Foucault:

> Το άτομο που καθυποβάλλεται σ' ένα πεδίο ορατότητας, και που το ξέρει, επωμίζεται το ίδιο τους καταναγκασμούς της εξουσίας· τους προσαρμόζει αυθόρμητα στον εαυτό του· δέχεται μέσα του τη σχέση εξουσίας όπου παίζει ταυτόχρονα και τους δύο ρόλους· γίνεται η βάση της ίδιας του της καθυπόταξης. (στο [15], σελ. 268)

Το [21] αναφέρει ότι το εν λόγω φαινόμενο συνέβη διαπιστωμένα στη Νορβηγία σε μια περίοδο τεσσάρων με πέντε δεκαετιών, ως αποτέλεσμα της παρακολούθησης των αριστερών από τις μυστικές υπηρεσίες.

Να σημειωθεί εδώ πως, όπως αναφέρεται στο [33], φαίνεται πως παραβίαση της ιδιωτικότητας μπορεί να υπάρξει ακόμη και με ψευδείς πληροφορίες: ας υποθέσουμε πως ένας πάροχος υγείας δημοσιεύει δεδομένα στα οποία το όνομα της Α σχετίζεται με κάποιες ψευδείς πληροφορίες για αυτήν[1]· τότε η Α έχει κάθε δικαίωμα να διαμαρτύρεται για παραβίαση της ιδιωτικότητάς της, ακόμη και στην περίπτωση που η ψευδής πληροφορία έχει κατά τα άλλα θετικό αντίκτυπο πάνω της.

Επιπλέον, όπως αναλύει το [20], η παραβίαση της ιδιωτικότητας οξύνει και οξύνεται από τις κοινωνικές ανισότητες. Αφενός, η δόλια χρήση προσωπικών δεδομένων με τους τρόπους που περιγράφονται πιο πάνω μπορεί εν γένει να οξύνει τις ανισότητες. Αφετέρου, αυτοί που βρίσκονται πιο ψηλά στην κοινωνική πυραμίδα ή έχουν στα χέρια τους μεγαλύτερη εξουσία διαθέτουν εν γένει περισσότερα μέσα για να παραβιάσουν την ιδιωτικότητα των άλλων και να προστατεύσουν τη δική τους: αποδοτικότερες ή πιο καινοτόμες τεχνολογίες, πρόσβαση σε πιο προστατευμένα περιβάλλοντα (πχ το γραφείο του διευθυντή σε μια επιχείρηση)· το πιο ακραίο παράδειγμα είναι το κράτος, που μπορεί να επιτρέψει –σε μεγάλο βαθμό– στον εαυτό του να παραβιάσει την ιδιωτικότητα των πολιτών, αλλά μπορεί να κάνει οποιοδήποτε εσωτερικό του έγγραφο απόρρητο και άρα απροσπέλαστο. Παρατηρείται άλλωστε πως οι περισσότερες περιπτώσεις στις οποίες εμφανίζεται ανάγκη για θέσπιση κανόνων προστασίας της ιδιωτικότητας είναι περιπτώσεις στις οποίες η μία πλευρά χρησιμοποιεί την εξουσία της για να καταπατήσει την ιδιωτικότητα της πιο αδύναμης πλευράς (πχ κράτος/αστυνομία έναντι πολιτών, εταιρείες/επαγγελματίες έναντι πελατών, φύλακες έναντι κρατουμένων).

Προφανώς, τα παραπάνω δεν ισχύουν οικουμενικά· η πολυπλοκότητα των κοινωνικών σχέσεων δημιουργεί περιστάσεις όπου η γενική τάση αντιστρέφεται. Στο [20] αναφέρεται για παράδειγμα το ότι κάποια άτομα που βρίσκονται πολύ ψηλά στην κοινωνική πυραμίδα παρακολουθούνται (έστω δυνητικά) συνεχώς, είτε ως άμεσο αποτέλεσμα της εξουσίας τους (πχ πολιτικοί)[2], είτε ως αποτέλεσμα των κοινωνικών σχέσεων που συνάπτουν με άλλα άτομα (πχ η υπηρέτρια μπορεί να παρακολουθεί τις ιδιωτικές στιγμές των αφεντικών της)· στην ίδια πηγή αναφέρεται ότι άτομα που βρίσκονται πολύ χαμηλά στην κοινωνική πυραμίδα επηρεάζονται λιγότερο από ορισμέ-

[1] Ας πούμε, για παράδειγμα, ότι η Α είναι μια διάσημη τραγουδίστρια και η ψευδής πληροφορία είναι ότι είναι δωρήτρια μυελού των οστών.

[2] Βέβαια, η παρακολούθηση δεν είναι ικανή συνθήκη για την παραβίαση της ιδιωτικότητας. Όπως σημειώνει το [21], η μαζική παρακολούθηση ατόμων με ισχύ γίνεται συνήθως διαμέσου των μέσων ενημέρωσης και χρησιμοποιείται για να παράξει ή να ενισχύσει πρότυπα συμπεριφοράς.

		Alice	
		Σ	Π
Bob	Σ	5,5	2,2
	Π	2,2	2,2

Σχήμα 1.1: Συνθέτοντας δύο επιμέρους παίγνια, το [37] καταλήγει στο παίγνιο που φαίνεται στον πίνακα. Η Alice και ο Bob διαθέτουν από ένα μυστικό (τη στρατηγική που θα ακολουθήσουν σε ένα υποκείμενο παίγνιο) και έκαστος έχει τη δυνατότητα να σεβαστεί (Σ) το μυστικό του αντιπάλου ή να το παραβιάσει (Π). Σε κάθε κελί του πίνακα, έχουμε το όφελος του Bob και το όφελος της Alice από κάθε συνδυασμό στρατηγικών, όπως προκύπτουν από τις ισορροπίες Nash στο υποκείμενο παίγνιο για τις επιλεγμένες στρατηγικές. Παρατηρούμε ότι τα οφέλη και των δύο μεγιστοποιούνται αν αμφότεροι επιλέξουν να σεβαστούν το μυστικό του αντιπάλου.

νες κατηγορίες δυνητικών παραβιάσεων της ιδιωτικότητάς τους (πχ κάποιος που δεν μπορεί να αγοράσει κινητό τηλέφωνο δεν μπορεί να παρακολουθηθεί μέσω αυτού). Τέτοια παραδείγματα υποδεικνύουν ότι υπάρχουν περιστάσεις στις οποίες η προστασία της ιδιωτικότητας ωφελεί όλους τους εμπλεκόμενους.

Μερικές κλάσεις τέτοιων περιστάσεων διερευνά το [37], χρησιμοποιώντας εργαλεία της θεωρίας παιγνίων. Θεωρεί παίγνια όπως του σχήματος 1.1 όπου τουλάχιστον ένας παίκτης έχει ένα μυστικό (το μυστικό μοντελοποιείται ως ψευδοτυχαιότητα στη στρατηγική του) του οποίου η διατήρηση οδηγεί σε ισορροπία Nash ή (ακόμη περισσότερο) μεγιστοποιεί το όφελος *όλων* των παικτών· κάτι τέτοιο μπορεί να απαιτεί –όπως και στην πραγματική ζωή– μηχανισμούς τιμωρίας ή ανωνυμίας. Όπως παραδέχεται και ο συγγραφέας, η εν λόγω προσέγγιση δεν μπορεί να θεωρηθεί οικουμενική[3], διότι θέτει μόνο οικονομικά κριτήρια για τον καθορισμό των περιπτώσεων όπου είναι επιθυμητή η προστασία της ιδιωτικότητας και έτσι θεωρεί ιδιωτικές μόνο τις πληροφορίες εκείνες των οποίων ο διαμοιρασμός είναι επιβλαβής για όλους τους εμπλεκόμενους (εκείνες δηλαδή που θα μπορούσαμε να πούμε πως έχουν «αρνητική αξία»).

Από τα προηγούμενα, προκύπτει πως η ιδιωτικότητα έχει μεγάλη σχέση με τη μυστικότητα, αλλά και με την ασφάλεια· οι τρεις έννοιες όμως δεν ταυτίζονται. Κατ' αρχάς, πολλές φορές η ασφάλεια και η ιδιωτικότητα έρχονται σε αντίθεση. Όπως αναφέρεται στο [39], υπάρχει περίπτωση για λόγους ασφαλείας να καταγράφονται αυτόματα οι δραστηριότητες σε ένα σύστημα, κάτι που προφανώς μπορεί να παραβιάσει την ιδιωτικότητα των χρηστών του· το [5] επιχειρηματολογεί πάνω στο πώς διαφέρουν από επιχειρησιακή σκοπιά οι δύο έννοιες και μάλιστα δίνει στα φαινόμενα που ο στόχοι τους αντιτίθενται (όπως στο προηγούμενο παράδειγμα) το –εντυπωσιακό– όνομα «παράδοξο ασφάλειας-ιδιωτικότητας». Κατά δεύτερον, η ιδιωτικότητα υποβοηθείται από τη μυστικότητα, αλλά δεν την απαιτεί. Είναι γεγονός (παραδείγματα δίνονται στο [32]) πως μπορούμε να έχουμε απαίτηση ιδιωτικότητας ακόμη και όταν βρισκόμαστε σε δημόσιο χώρο ή όταν είναι γνωστό το τι ακριβώς κάνουμε· χρησιμοποιώντας τα λόγια του Floridi:

[3]Εκτός βέβαια αν υιοθετήσουμε έναν (αφελή) ακραίο ωφελιμισμό.

> Στις ανθρώπινες κοινωνίες, η ιδιωτικότητα προστατεύεται και μέσω σιωπηλών συμφωνιών. Αγνοούμε «από ευγένεια» –για παράδειγμα δεν αναφέρουμε– στιγμές που όλοι ζούμε και γνωρίζουμε, ευγένεια που εκτείνεται από το ότι κοιτάμε ευθεία μπροστά στα ουρητήρια μέχρι το ότι δεν μιλάμε για συζυγικές πράξεις που ξέρουμε (και μερικές φορές έχουμε αποδείξεις) ότι συμβαίνουν. Καμία θεωρία για την πληροφοριακή ιδιωτικότητα δεν είναι πλήρης αν δεν μπορεί να αντιμετωπίσει τέτοια φαινόμενα.[4]

Ήδη από το 1975, ο Rachels υποστήριξε [31] πως η ιδιωτικότητα είναι απαραίτητο στοιχείο για τη δόμηση και τη διατήρηση υγιών κοινωνικών σχέσεων, διότι η βιωσιμότητα μιας κοινωνικής σχέσης εξαρτάται από το διαμοιρασμό κάποιων πληροφοριών και τον μη διαμοιρασμό κάποιων άλλων. Η θεωρία του έχει δεχθεί αρκετή κριτική. Για παράδειγμα, το [24] σημειώνει πως υπάρχουν άλλοι αρκετά σημαντικοί παράγοντες στις ανθρώπινες σχέσεις, όπως η φυσική παρουσία, ο τρόπος που γίνεται ο διαμοιρασμός της πληροφορίας, η στοργή κ.ά.· επιπλέον, η ίδια πηγή ισχυρίζεται πως η αυξημένη διαθεσιμότητα των πληροφοριών που παρατηρείται στις μέρες μας σε σχέση με παλιότερα δεν φαίνεται να οδηγεί σε αντίστοιχη απορρύθμιση των κοινωνικών σχέσεων. Παρόλα αυτά, δέχεται πως η θεωρία του Rachels εξακολουθεί να δίνει κάποιες ενδιαφέρουσες κατευθύνσεις σε σχέση με τον σχεδιασμό κοινωνικών δικτύων, καθώς και με το ζήτημα της παρακολούθησης.

Να σημειωθεί εδώ ότι, όπως αναγνωρίζει και το [14], η ιδιωτικότητα δεν έχει την ίδια αξία ή τα ίδια χαρακτηριστικά σε όλους τους πολιτισμούς, κάτι που μερικές φορές δεν λαμβάνεται υπόψιν στις προσπάθειες προσέγγισής της.

1.2 Ο ρόλος των νέων τεχνολογιών

Δεν είναι δύσκολο να πείσουμε τον εαυτό μας ότι οι νέες τεχνολογίες που σχετίζονται με την καταγραφή και μετάδοση πληροφορίας έχουν συνέπειες στην ιδιωτικότητα. Άλλωστε, σύμφωνα με το [10], η πρώτη σύγχρονη υπεράσπιση (εν έτει 1890) της ιδιωτικότητας ως έχουσας ανάγκη (νομικής) προστασίας [36] επηρεάστηκε σε μεγάλο βαθμό από την διάχυση τέτοιων τεχνολογιών, εν προκειμένω των εφημερίδων και της φωτογραφίας. Πλήθος πηγών (ενδεικτικά [2, 9, 27]) αναλύουν τον τρόπο που η ευρεία χρήση των υπολογιστών και του διαδικτύου διαμορφώνει νέες κοινωνικές συνθήκες στις μέρες μας.

Θα μπορούσε κανείς να ισχυριστεί ότι οι νέες τεχνολογίες, διευκολύνοντας την παρακολούθηση, δίνουν την ευκαιρία στους ανίσχυρους να παρακολουθούν και αυτοί τους ισχυρούς, προστατεύοντας έτσι καλύτερα τα συμφέροντά τους. Η ουδετερότητα όμως της δυνητικά αμφίδρομης παρακολούθησης είναι, όπως επιχειρηματολογεί το [20], φαινομενική, διότι μπορεί να είναι –και σε πολλές περιπτώσεις όντως είναι– ασύμμετρη, με την πλευρά

[4] In human societies privacy is also fostered through tacit agreements. We «politely» ignore –e.g., do not bring up in conversation– moments we all witness and know about, ranging from keeping our eyes straight ahead at the urinal to never speaking of, say, marital acts that we know (and sometimes have evidence to confirm) must take place. [N]o theory of informational privacy is complete that cannot account for such phenomena [14].

που έχει μεγαλύτερη εξουσία να επωφελείται περισσότερο· είναι απόλυτα αναμενόμενο τα μέσα που τελικά είναι διαθέσιμα στην αδύναμη πλευρά να είναι λιγότερα, είτε λόγω μεγάλου κόστους (σε χρήμα ή χρόνο) είτε λόγω επιβολής του αντιπάλου. Εξάλλου, συνήθως είναι οι ισχυροί που αργά ή γρήγορα ελέγχουν τον τρόπο που θα χρησιμοποιηθούν τα νέα μέσα (βλ. και υποσημείωση 2 στη σελίδα 2).

1.3 Ορισμοί και προσεγγίσεις της έννοιας της ιδιωτικότητας

Στη βιβλιογραφία, η ιδιωτικότητα εμφανίζεται με διάφορες μορφές:

- Η φυσική ιδιωτικότητα (physical privacy) αφορά την ανάγκη των ανθρώπων να έχουν τον δικό τους ιδιωτικό χώρο, στον οποίο να μην ενοχλούνται ή παρατηρούνται από άλλους.
- Η πληροφοριακή ιδιωτικότητα (informational privacy) αφορά τον τρόπο που συλλέγονται και διαμοιράζονται δεδομένα ή πληροφορίες σχετικές με άτομα.
- Η ιδιωτικότητα στη λήψη αποφάσεων (decisional privacy) αφορά το δικαίωμα στη λήψη αποφάσεων χωρίς ενοχλήσεις από τρίτους.
- Η συνταγματική ιδιωτικότητα (constitutional privacy) αφορά το δικαίωμα στην ιδιωτικότητα όπως ορίζεται από τους νόμους.

1.3.1 Η πληροφοριακή οντολογία του Floridi

Ο Floridi [14] υπερασπίζεται την πληροφοριακή ιδιωτικότητα ως βασικό στοιχείο της ανθρώπινης αξιοπρέπειας, μέσα σε ένα οντολογικό φιλοσοφικό πλαίσιο.

Συγκεκριμένα, ο Floridi χρησιμοποιεί ένα οντολογικό σχήμα σύμφωνα με το οποίο οι πληροφορίες (όχι κατ' ανάγκη ψηφιακές) και οι φορείς τους σχηματίζουν ένα «σύμπαν», την πληροφοριόσφαιρα (infosphere). Η ροή των πληροφοριών μέσα στην πληροφοριόσφαιρα δυσχεραίνεται από διάφορες οντολογικές ιδιότητές της (χρόνος, χώρος, διαθέσιμοι πόροι, πολυπλοκότητα δεδομένων), οι οποίες αυξάνουν την απαιτούμενη προσπάθεια για τη απόκτηση, το φιλτράρισμα ή την παρεμπόδιση της πληροφορίας· αναφερόμαστε στους παράγοντες αυτούς με τον όρο οντολογική τριβή (ontological friction). Να σημειωθεί εδώ πως η οντολογική τριβή επηρεάζει με διαφορετικό τρόπο το κάθε άτομο, ανάλογα με τις πληροφοριακές του δυνατότητες.

Στα πλαίσια της πληροφοριόσφαιρας, κάθε άνθρωπος αποτελείται από τις σχετικές με αυτόν πληροφορίες και η ιδιωτικότητά του ορίζεται ως ουσιαστικό στοιχείο της αξιοπρέπειας και της ακεραιότητάς του· συγκεκριμένα, ιδιωτικότητα είναι το δικαίωμα να μην δέχεται ανεπιθύμητες ή άγνωστες, ενεργητικές ή παθητικές, μεταβολές στην ταυτότητά του ως πληροφοριακή οντότητα. Με αυτή την έννοια, η συλλογή, η αποθήκευση, η επεξεργασία και ο διαμοιρασμός πληροφοριών για ένα άτομο, είναι προσβολές στην ιδιωτικότητά του, διότι (και στο βαθμό που) ανταποκρίνονται σε αντίστοιχες

ενεργητικές πράξεις στην ταυτότητά του· ομοίως, η έκθεσή του σε ανεπιθύμητα δεδομένα είναι ενδεχομένως μια παθητική αλλοίωση της ταυτότητάς του, και επομένως προσβολή της ιδιωτικότητάς του. Είναι προφανές πως η ιδιωτικότητα σε ένα τέτοιο πλαίσιο εξαρτάται από την οντολογική τριβή.

Η πληροφοριακή οντολογία του Floridi αναγνωρίζει στις νέες τεχνολογίες τον κεντρικό τους ρόλο σχετικά με την ιδιωτικότητα. Συγκεκριμένα, οι νέες τεχνολογίες πληροφορίας ή επικοινωνίας (ΤΠΕ) επηρεάζουν την οντολογική τριβή, επομένως έμμεσα και την ιδιωτικότητα. Ο Floridi κάνει διάκριση ανάμεσα στις παλιές και τις νέες ΤΠΕ: οι παλιές μειώνουν την οντολογική τριβή –άρα αυξάνουν την απαιτούμενη προσπάθεια για προστασία της ιδιωτικότητας–, ενώ οι νέες ΤΠΕ μεταμορφώνουν ολόκληρη την οντολογία της πληροφοριόσφαιρας, διευκολύνοντας την προστασία της ιδιωτικότητας σε ορισμένες περιστάσεις, δυσχεραίνοντάς την σε άλλες και ενδεχομένως τροποποιώντας την ίδια την έννοια.

1.3.2 Η ιδιωτικότητα ως δικαίωμα και η θεωρία των φραγμών του Rickless

Πολλές προσεγγίσεις της ιδιωτικότητας έχουν γίνει από την μεριά της ηθικής ή της νομικής, όπου η ιδιωτικότητα θεωρείται κάποιου είδους δικαίωμα και σκοπός είναι η διασαφήνισή του. Ο Rickless [32] κάνει μία ταξινόμηση των διάφορων προσεγγίσεων, σε δύο επίπεδα: στο πρώτο επίπεδο, υπάρχει η αντιδιαστολή ανάμεσα στους αναγωγιστές (reductionists), οι οποίοι θεωρούν πως η ιδιωτικότητα δεν είναι ένα αυθύπαρκτο δικαίωμα, αλλά αποτελεί ειδική περίπτωση ενός ή περισσότερων άλλων δικαιωμάτων (πχ στην περιουσία), και στους μη αναγωγιστές (non-reductionists), οι οποίοι θεωρούν την ιδιωτικότητα αυθύπαρκτο δικαίωμα. Σε δεύτερο επίπεδο, οι μη αναγωγιστές χωρίζονται σε αυτούς που θεωρούν την ιδιωτικότητα ως ένα δικαίωμα που έχει να κάνει με πρόσβαση τρίτων στις σχετικές με το άτομο πληροφορίες, σε αυτούς που θεωρούν την ιδιωτικότητα ως ένα δικαίωμα που έχει να κάνει με τον έλεγχο του ατόμου πάνω στις πληροφορίες του και σε αυτούς που υιοθετούν μικτές προσεγγίσεις.

Χρησιμοποιώντας μια σειρά από παραδείγματα, ο Rickless κάνει κριτική σε προγενέστερες θεωρίες όλων των ειδών που αναφέρθηκαν και διατυπώνει τη δική του, που αποκαλεί θεωρία των φραγμών (barrier theory), η οποία ξεπερνάει τις δυσκολίες που παρουσιάζονται στις προγενέστερες θεωρίες. Στη θεωρία των φραγμών, η ιδιωτικότητα ορίζεται ως εξής:

> Ο X έχει δικαίωμα στην ιδιωτικότητα έναντι του Y αν [ο Y έχει υποχρέωση απέναντι στον X πως] δεν πρέπει να μάθει ή να βιώσει κάποιο προσωπικό γεγονός σχετικά με τον X παραβιάζοντας ένα φραγμό που χρησιμοποιείται από τον X ώστε να εμποδίσει άλλους από το να μάθουν ή να βιώσουν κάποιο προσωπικό γεγονός σχετικά με τον X[5].

Ο φραγμός που χρησιμοποιεί ο X δεν χρειάζεται να είναι φυσικό αντικείμενο (μπορεί να είναι απλά η απόσταση) και δεν χρειάζεται να ανήκει στον

[5]For X to have right to privacy against Y is for [Y to be under a duty toward X] that Y not learn or experience some personal fact about X by breaching a barrier used by X to keep others from learning or experiencing some personal fact about X.

Συλλογή πληροφοριών	**Επεξεργασία πληροφοριών**	**Διαμοιρασμός πληροφοριών**	**Επέμβαση**
Παρακολούθηση Ανάκριση	Συγκέντρωση Ταυτοποίηση Ανασφάλεια Δευτερεύουσες χρήσεις Αποκλεισμός	Παραβίαση της εμπιστευτικότητας Κοινοποίηση Έκθεση Αυξημένη προσβασιμότητα Εκβιασμός Οικειοποίηση Διαστρέβλωση	Παρείσφρηση Παρεμβολή στη λήψη απόφασης

Σχήμα 1.2: Η ταξινομία του Solove

X, ούτε να χρησιμοποιείται συνειδητά. Σύμφωνα με τον ορισμό, η απλή παραβίαση του φραγμού δεν συνιστά παραβίαση της ιδιωτικότητας· όπως επιχειρηματολογεί ο Rickless, αν κάποιος διαθέτει μια συσκευή ακτίνων Χ που του επιτρέπει να δει το εσωτερικό κάθε σπιτιού, αλλά δεν την έχει χρησιμοποιήσει ποτέ, τότε δεν έχει παραβιάσει την ιδιωτικότητα κανενός. Να σημειωθεί εδώ πως ο Rickless θεωρεί ότι όταν ο X γνωρίζει πως ο φραγμός δεν μπορεί να εμποδίσει ακόμη και καλόβουλους τρίτους (για παράδειγμα, κάποιος που απλά στέκεται στο διάδρομο μπορεί να ακούσει τι λέει ο X μέσα από την πόρτα του διαμερίσματός του), τότε έχει παραιτηθεί από το δικαίωμά του στην ιδιωτικότητα.

1.3.3 Η ταξινομία του Solove

Ο Solove στο [34], όντας πεπεισμένος ότι η ιδιωτικότητα δεν έχει μία ενιαία ουσία («singular essence»), αλλά εγκολπώνει επιμέρους έννοιες συνδεόμενες με σχέσεις «οικογενειακής ομοιότητας» (με την έννοια που χρησιμοποιεί τον όρο ο Wittgenstein) και ορμώμενος από την ανάγκη για σαφή νομικό καθορισμό της ιδιωτικότητας, προσφέρει μία ταξινομία των διάφορων πράξεων που μπορούν να οδηγήσουν σε παραβίασή της, δίνοντας μάλιστα παραδείγματα από πραγματικές περιπτώσεις που έχουν απασχολήσει τα δικαστήρια. Όπως σημειώνει και ο ίδιος, η ταξινομία του μπορεί να είναι χρήσιμο εργαλείο, αλλά δεν αποτελεί προσπάθεια ορισμού της ιδιωτικότητας. Θα περιγράψουμε τα στοιχεία της ταξινομίας αυτής, η οποία φαίνεται συνοπτικά στο σχήμα 1.2 και αποτελείται από τέσσερις κατηγορίες:

- Η πρώτη κατηγορία περιλαμβάνει τις πράξεις που συμβαίνουν κατά τη συλλογή πληροφοριών. Αυτές είναι
  - η παρακολούθηση, η οποία συνίσταται στην παρατήρηση ή/και καταγραφή (οπτική, ακουστική) των δραστηριοτήτων ενός ατόμου ή πληροφοριών σχετικά με αυτό,
  - η ανάκριση, η οποία συνίσταται στην προσπάθεια για λήψη πληροφοριών από το άτομο, αλληλεπιδρώντας με αυτό (για παράδειγμα κάνοντάς του ερωτήσεις).

- Η δεύτερη κατηγορία περιλαμβάνει τις πράξεις που συμβαίνουν κατά την επεξεργασία, αποθήκευση ή χρήση των πληροφοριών.
  - Η συγκέντρωση είναι ο συνδυασμός διαφορετικών πληροφοριών με αποτέλεσμα τη συναγωγή καινούριας πληροφορίας.
  - Η ταυτοποίηση είναι η χρήση των πληροφοριών που καταλήγει στην αποκάλυψη της (μέχρι τότε άγνωστης) ταυτότητας ατόμων.
  - Η ανασφάλεια αφορά τις αβλεψίες στην αποθήκευση και μεταφορά πληροφοριών που μπορούν να επιτρέψουν ή να διευκολύνουν διαρροή τους.
  - Οι δευτερεύουσες χρήσεις είναι η χρήση των δεδομένων για σκοπούς άλλους από αυτούς με βάση τους οποίους συλλέχθηκαν.
  - Αποκλεισμός καλείται η αδυναμία του ατόμου το οποίο αφορούν οι πληροφορίες να έχει πρόσβαση σε αυτές και τη χρήση τους.
- Η τρίτη κατηγορία περιλαμβάνει τις πράξεις που συμβαίνουν κατά το διαμοιρασμό πληροφοριών και είναι η πολυπληθέστερη:
  - Η παραβίαση της εμπιστευτικότητας είναι η αθέτηση της υπόσχεσης ή της υποχρέωσης να διατηρηθούν μυστικές κάποιες πληροφορίες που αποκτώνται λόγω της σχέσης (πχ επαγγελματικής) με ένα άτομο.
  - Η κοινοποίηση αφορά την αποκάλυψη αληθών πληροφοριών που μπορούν να οδηγήσουν σε κρίσεις για το χαρακτήρα ενός ατόμου.
  - Η έκθεση είναι η αποκάλυψη πληροφοριών που θεωρούνται ταμπού, όπως γυμνές φωτογραφίες.
  - Η αυξημένη προσβασιμότητα είναι η διευκόλυνση τρίτων στην απόκτηση πληροφοριών.
  - Ο εκβιασμός είναι η απειλή για αποκάλυψη προσωπικών δεδομένων.
  - Η οικειοποίηση είναι η χρήση των δεδομένων ενός ατόμου για την προώθηση των συμφερόντων ενός τρίτου.
  - Τέλος, η διαστρέβλωση είναι η διασπορά ψευδών πληροφοριών.
- Η τέταρτη κατηγορία περιλαμβάνει τις πράξεις που επεμβαίνουν κατ' ευθείαν στο άτομο και είναι η μόνη που δεν αφορά αποκλειστικά την πληροφοριακή ιδιωτικότητα.
  - Η παρείσφρηση αφορά την παρέμβαση στην ίδια τη ζωή του ατόμου, που μπορεί να έχει αποτέλεσμα την ενόχληση ή την αλλαγή των συνηθειών του. Πολλές φορές εμφανίζεται μαζί με την συλλογή πληροφοριών, αλλά έχει πιο βαθιές συνέπειες.
  - Η παρεμβολή στη λήψη απόφασης είναι η δημιουργία συνθηκών (συνήθως από το κράτος) που δυσκολεύουν ή απαγορεύουν τη λήψη αποφάσεων σχετικά με ιδιωτικά ζητήματα.

1.3.4 Ο τέλειος ηδονοβλεψίας

Στο [12], ο Doyle εισάγει το ενδιαφέρον παράδειγμα του τέλειου ηδονοβλεψία. Χρησιμοποιώντας τα δικά του λόγια:

Παράδειγμα 1.1 (Ο τέλειος ηδονοβλεψίας). Φανταστείτε τον στα βάθη της Μογγολίας, στον Άρη ή σε απόσταση 100 ετών φωτός. Έχει πλήρη και μη αντιληπτή οπτικοακουστική εποπτεία στο διαμέρισμα, τους υπολογιστές και τα τηλέφωνά της. Επιπλέον, γνωρίζει το πλήρες ιατρικό ιστορικό και το γονιδίωμά της. Επομένως, το θύμα του έχει απωλέσει πλήρως την ατομική και πληροφοριακή της ιδιωτικότητα [...]. Παρόλα αυτά, δεν έρχεται ποτέ σε οποιαδήποτε επαφή μαζί της, άρα δεν μπορεί να τη βλάψει άμεσα. Δεν τη βλάπτει ούτε έμμεσα, αμαυρώνοντας τη φήμη της ή μοιραζόμενος με οποιονδήποτε άλλον τις παρατηρήσεις του. Εν συντομία, αφήνει αυτή και τον υπόλοιπο κόσμο ανεπηρέαστους. Ούτε η ελεεινή συνήθειά του διαβρώνει τον δικό του χαρακτήρα. Είναι βράχος. Η ηδονοβλεψία του είναι τέλεια: απαρατήρητη και αγνωστοποίητη [...].[6]

Να σημειωθεί εδώ πως ο Doyle υπερασπίζεται μία ωφελιμιστική ερμηνεία της ιδιωτικότητας: η απώλειά της είναι ανήθικη επειδή προκαλεί ζημιά σε αυτόν που τη χάνει, είτε άμεση (πχ εκβιασμός, αλλαγή συμπεριφοράς του υποκειμένου λόγω γνώσης ότι παρακολουθείται) είτε έμμεση (πχ αμαύρωση της φήμης του υποκειμένου, χρήση των πληροφοριών για προβάδισμα), η οποία υπερβαίνει την όποια ευχαρίστηση του καταπατητή. Σύμφωνα λοιπόν με τον Doyle, από τη στιγμή που ο τέλειος ηδονοβλεψίας δεν προκαλεί καμία άμεση ή έμμεση ζημιά στο θύμα του, η πράξη του δεν είναι κατακριτέα. Η ενδεχόμενη ζημιά που προκαλεί στο θύμα καταπατώντας την επιθυμία του να μην παρακολουθείται αντισταθμίζεται από τη δική του επιθυμία να παρακολουθεί. Αντίθετα, αν ο ίδιος έχει κάποια ικανοποίηση από την παρακολούθηση, τότε η ύπαρξή του κάνει τον κόσμο καλύτερο και άρα είναι επιθυμητή· με το ίδιο σκεπτικό, καταλήγουμε ότι ένας κόσμος με $n+1$ ανεξάρτητους τέλειους ηδονοβλεψίες είναι καλύτερος από έναν κόσμο με n ανεξάρτητους ηδονοβλεψίες.

Παρατηρούμε εδώ ότι ο Doyle έχει κατασκευάσει ένα παράδειγμα όπου (αν προσπαθήσουμε να δούμε το πρόβλημα έξω από την ωφελιμιστική σκοπιά) το θύμα έχει απωλέσει πλήρως την ιδιωτικότητά του, αλλά μπορεί να έχει την αυτονομία του, με την έννοια ότι οι επιλογές και τα ερεθίσματά του είναι τα ίδια με ή χωρίς την παρουσία του τέλειου ηδονοβλεψία. Έτσι, σημειώνει πως η ιδιωτικότητα δεν είναι ικανή συνθήκη για την αυτονομία.

[6]The perfect voyeur: Stick him in Outer Mongolia, Mars, or 100 light years away. He's got her apartment, computers, and phones under complete and undetectable audio and visual surveillance. Moreover, he knows her full medical history and her genetic makeup. So his victim utterly lacks both personal and informational privacy [...]. However, he never has any contact with her, so he never directly harms her. Nor does he indirectly harm her by sullying her reputation or clueing others in on the show. In short, he leaves her alone, and so does everyone else. Neither does his nasty habit corrupt other aspects of his character. He is a pillar. His voyeurism is perfect: undetected and unpublicized [...].

1.4 Προσπάθειες προστασίας της ιδιωτικότητας

1.4.1 Το νομικό πλαίσιο στην Ευρωπαϊκή Ένωση και την Ελλάδα

Τον Ιούλιο του 1990, η Ευρωπαϊκή Κοινότητα εξέδωσε την πρώτη πρόχειρη πρόταση για μία οδηγία περί της προστασίας των προσωπικών δεδομένων. Μετά από χρόνια διαβουλεύσεων, η τελική Οδηγία 95/46/ΕΚ «για την προστασία των φυσικών προσώπων έναντι της επεξεργασίας δεδομένων προσωπικού χαρακτήρα και για την ελεύθερη διακίνηση των δεδομένων αυτών» υιοθετήθηκε από το Ευρωπαϊκό Συμβούλιο το 1995. Η Οδηγία αυτή αποτέλεσε ορόσημο και αποτέλεσε τη βάση για τη νομοθεσία όχι μόνο των κρατών–μελών της Ευρωπαϊκής Κοινότητας αλλά σε παγκόσμιο επίπεδο. Από τότε, εκδόθηκαν πλείστες άλλες Οδηγίες για να διευθετήσουν ζητήματα που αφορούσαν τις τεχνολογικές εξελίξεις. [39]

Το 2016, εκδόθηκε ο Κανονισμός (ΕΕ) 2016/679, ο οποίος αντικαθιστά την Οδηγία 95/46/ΕΚ και θα υιοθετηθεί από τα κράτη-μέλη μέχρι το 2018 [38]. Επίσης, εγκρίθηκε η ασπίδα προστασίας ΕΕ-ΗΠΑ για την ιδιωτικότητα, μια διμερής συμφωνία που ορίζει διαδικασίες που πρέπει να ακολουθούνται όταν δεδομένα Ευρωπαίων πολιτών μεταφέρονται σε εταιρείες στις ΗΠΑ [13].

Η Ελλάδα κατοχυρώνει συνταγματικά ως ατομικά και κοινωνικά δικαιώματα το δικαίωμα στην προστασία προσωπικών δεδομένων (άρθρο 9) και στο απόρρητο της επικοινωνίας (άρθρο 19). Επιπλέον, όπως είναι επόμενο, η Ελλάδα έχει ενσωματώσει τις Κοινοτικές Οδηγίες που περιγράφονται παραπάνω. Στο πλαίσιο αυτό, η Οδηγία 95/46/ΕΚ τέθηκε σε ισχύ τον Απρίλιο του 1997 με το Νόμο 2472/1997, με τον οποίο συγκροτήθηκε ως ανεξάρτητη δημόσια αρχή η Αρχή Προστασίας Δεδομένων Προσωπικού Χαρακτήρα (ΑΠΔΠΧ), που έχει ως αποστολή την εποπτεία της εφαρμογής του νόμου και άλλων ρυθμίσεων που αφορούν την προστασία του ατόμου από την επεξεργασία δεδομένων προσωπικού χαρακτήρα, καθώς και την ενάσκηση των κατά περίπτωση αρμοδιοτήτων που της ανατίθενται. Πέρα από την ΑΠΔΠΧ, ο Νόμος 3115/2003 συγκρότησε μία δεύτερη ανεξάρτητη αρχή, την Αρχή Διασφάλισης Απορρήτου των Επικοινωνιών (ΑΔΑΕ), με σκοπό την «προστασία του απορρήτου των επιστολών, της ελεύθερης ανταπόκρισης ή επικοινωνίας με οποιονδήποτε άλλο τρόπο καθώς και [την] ασφάλεια των δικτύων και πληροφοριών». Ο Νόμος 3471/2006 ενσωματώνει την Οδηγία 2002/58/ΕΚ και τροποποιεί το Νόμο 2472/97 ως προς ορισμένες διατάξεις του, ενώ ο Νόμος 3917/2011 αποτελεί την υλοποίηση της Οδηγίας 2006/24/ΕΚ και πραγματεύεται τη διατήρηση δεδομένων που παράγονται ή υποβάλλονται σε επεξεργασία σε συνάρτηση με την παροχή διαθέσιμων στο κοινό υπηρεσιών ηλεκτρονικών επικοινωνιών ή δημόσιων δικτύων επικοινωνιών, χρήση συστημάτων επιτήρησης με τη λήψη ή καταγραφή ήχου ή εικόνας σε δημόσιους χώρους και συναφή ζητήματα. [39]

1.4.2 Ιδιωτικότητα εκ σχεδιασμού

Όπως σημειώνεται στο [11], το γεγονός ότι υπάρχει (περιορισμένο έστω) νομικό πλαίσιο για την προστασία της ιδιωτικότητας δεν συνεπάγεται πως αυτή όντως προστατεύεται· ανάμεσα στις αιτίες μπορεί να είναι η απουσία

αυστηρών συνεπειών σε παραβάσεις, αλλά πολύ συχνά είναι η απουσία εργαλείων και μεθόδων που να βοηθούν όντως στην προστασία των δεδομένων. Στην ίδια πηγή αναφέρονται κάποια παραδείγματα πρακτικών προβλημάτων στην ανάπτυξη λογισμικού και λύσεών τους που σέβονται την ιδιωτικότητα. Στο ίδιο πνεύμα κινείται η έννοια της ιδιωτικότητας εκ σχεδιασμού (privacy by design) [6], που αποτελεί ένα σύνολο αρχών για την ενσωμάτωση των απαιτήσεων ιδιωτικότητας στο σχεδιασμό και τη λειτουργία του λογισμικού.

Σε αυτή την κατεύθυνση μπορούν να δώσουν ισχυρή ώθηση οι τυπικές μέθοδοι, δηλαδή μέθοδοι για τη μαθηματική μελέτη ιδιοτήτων προγραμμάτων. Μπορούν να χρησιμοποιηθούν για να παράξουν κώδικα από δεδομένες προδιαγραφές, για να ελέγξουν –στατικά ή δυναμικά– τη συμμόρφωση προγραμμάτων με πολιτικές ιδιωτικότητας (και άρα ενδεχομένως να πιστοποιήσουν προγράμματα), καθώς και για να εφοδιάσουν τους σχεδιαστές λογισμικού με ορθά μοντέλα περί ιδιωτικότητας [18].

Κεφάλαιο 2

Έλεγχος τύπων, η Maude και τα θεμέλιά της

2.1 Τι είναι και πού στοχεύει ο έλεγχος τύπων

Ένα από τα καίρια ζητήματα της σχεδίασης και συγγραφής λογισμικού είναι το κατά πόσο αυτό ικανοποιεί τις προδιαγραφές που τίθενται (ρητά ή όχι). Προφανώς, το πρόβλημα αυτό είναι μη επιλύσιμο στην γενική περίπτωση· αυτό έχει δώσει ώθηση στο ευρύ πεδίο που είναι γνωστό ως Τυπικές Μέθοδοι. Το πεδίο αυτό περιλαμβάνει τεχνικές που μπορούν να χρησιμοποιηθούν, είτε αναλύοντας τον κώδικά του (στατικά) είτε κατά τη διάρκεια της εκτέλεσής του, για να διαπιστώσουμε κατά πόσο ένα συγκεκριμένο πρόγραμμα ικανοποιεί μία συγκεκριμένη προδιαγραφή. Μία δημοφιλής τέτοια τεχνική είναι ο έλεγχος τύπων. Σύμφωνα με το [30], από την εισαγωγή του οποίου προέρχονται οι πληροφορίες της παρούσας υποενότητας:

> Ένα σύστημα τύπων είναι μία υπολογιστικά εφικτή συντακτική μέθοδος για την απόδειξη της απουσίας συγκεκριμένων συμπεριφορών ενός προγράμματος, μέσω της ταξινόμησης εκφράσεων σε κλάσεις ανάλογα με το είδος των τιμών που υπολογίζουν.[1]

Όπως παραδέχεται και ο συγγραφέας του [30], ο ορισμός του συστήματος τύπων είναι δύσκολος και ο συγκεκριμένος παραμερίζει χρήσεις της τεχνικής σε πεδία ευρύτερα του προγραμματισμού (εξάλλου, ο Russell είχε ορίσει ένα μεταμαθηματικό σύστημα τύπων για να αποφύγει τα παράδοξα της θεωρίας συνόλων).

Ο τρόπος που δουλεύει ο έλεγχος τύπων είναι, όπως φαίνεται και από τον πιο πάνω ορισμό, να αναλύει εκφράσεις και με βάση τη δομή τους να συνάγει την κλάση (τον τύπο) στην οποία ανήκει το –οποιοδήποτε– αποτέλεσμα της εκτέλεσής τους. Κατά κανόνα, ο έλεγχος τύπων γίνεται κατά τη μεταγλώττιση ή τη σύνδεση ενός προγράμματος, άρα πρέπει να είναι σε θέση να λειτουργεί αποδοτικά και αυτόματα. Η αυτόματη λειτουργία του υποβοηθείται συνήθως από ενδείξεις που ενσωματώνει στον κώδικα ο προγραμματιστής.

[1] A type system is a tractable syntactic method for proving the absence of certain program behaviors by classifying phrases according to the kinds of values they compute.

Ο έλεγχος τύπων είναι συντηρητικός, με την έννοια ότι μπορεί να εντοπίσει μόνο την πλήρη απουσία ανεπιθύμητων συμπεριφορών· είναι όμως πιθανό κάποια συμπεριφορά να είναι λογικά δυνατή, αλλά, λόγω διάφορων συνθηκών που δεν μπορούν να εκφραστούν στο σύστημα τύπων που διαθέτουμε (ή ίσως θα ήταν υπολογιστικά δύσκολο να διαπιστώσουμε ότι ισχύουν), δεν πρόκειται να εμφανιστεί ποτέ στην πράξη. Επίσης, το σύστημα τύπων δεν αποφαίνεται για αυθαίρετες συμπεριφορές ενός προγράμματος, παρά μόνο για εκείνες που λαμβάνονται υπόψιν στο σχεδιασμό του. Οι συμπεριφορές ενός προγράμματος που μπορούν να εντοπιστούν με τον έλεγχο τύπων περιλαμβάνουν, ενδεικτικά, λάθη στη χρήση τύπων δεδομένων, αλλά και παραβιάσεις γενικότερων υποθέσεων σχετικών με την ακεραιότητα ή την ασφάλεια δεδομένων.

Εκτός από τον εντοπισμό σφαλμάτων, τα συστήματα τύπων βοηθούν σε πιο αφαιρετική αναπαράσταση της λειτουργίας του προγράμματος, στην ευκολότερη ανάγνωσή του, στον εντοπισμό κατά τη μεταγλώττιση ιδιοτήτων που μπορούν να οδηγήσουν σε ταχύτερη εκτέλεση, καθώς και στην αυτόματη απόδειξη θεωρημάτων.

2.2 Τα μαθηματικά θεμέλια της Maude

2.2.1 Συστήματα αναγωγών

Ένα σύστημα αναγωγών (reduction system) είναι ένα σύνολο, το οποίο μπορούμε να φανταστούμε ως σύνολο όρων μιας γλώσσας, εφοδιασμένο με μία διμελή σχέση που καλείται αναγωγή (reduction), την οποία μπορούμε να φανταστούμε ως ένα σύνολο κανόνων που οδηγούν από κάποιους όρους σε κάποιους άλλους.

Ορισμός 2.1. Αν σε ένα στοιχείο x ενός συστήματος αναγωγών δεν μπορεί να εφαρμοστεί καμία αναγωγή, τότε λέμε ότι το x βρίσκεται σε κανονική μορφή (normal form). Αν από ένα στοιχείο y οδηγούμαστε στο x με μια σειρά πεπερασμένων αναγωγών, λέμε ότι το x είναι κανονική μορφή του y. Αν κάθε στοιχείο έχει κανονική μορφή, τότε λέμε ότι το σύστημα αναγωγών τερματίζει.

Μία πολύ σημαντική ιδιότητα που χρησιμοποιεί η Maude είναι η ιδιότητα Church-Rosser. Επειδή η διατύπωση της ίδιας της ιδιότητας χρειάζεται μερικούς ακόμα ορισμούς, θα δώσουμε μια ισοδύναμη διατύπωσή της (περισσότερα μπορούν να βρεθούν στο [1]).

Ορισμός 2.2. Αν σε ένα σύστημα αναγωγών υπάρχει ένας όρος x στον οποίον μπορούν να γίνουν δύο (ενδεχομένως κενές) ακολουθίες αναγωγών, οδηγώντας στο y και στο z αντίστοιχα, τότε υπάρχουν ένα w και δύο (ενδεχομένως κενές) ακολουθίες αναγωγών ώστε η πρώτη να οδηγεί από το y στο w ενώ η δεύτερη να οδηγεί από το z στο w.

Η σημαντικότητα της ιδιότητας Church-Rosser έγκειται στο παρακάτω πόρισμα:

Συνέπεια 2.3. *Αν ένα σύστημα αναγωγών έχει την ιδιότητα Church-Rosser, η κανονική μορφή κάθε στοιχείου (αν υπάρχει) είναι μοναδική.*

Απόδειξη. Έστω ότι το στοιχείο x του συστήματος αναγωγών έχει κανονικές μορφές y_1 και y_2. Τότε, από την ιδιότητα Church-Rosser, υπάρχουν ένα w και δύο ακολουθίες αναγωγών ώστε η πρώτη να οδηγεί από το y_1 στο w ενώ η δεύτερη να οδηγεί από το y_2 στο w. Επειδή τα y_1 και y_2 είναι κανονικές μορφές, δεν υπάρχουν αναγωγές από αυτά· επομένως, $w = y_1 = y_2$. □

2.2.2 Η εξισωτική λογική

Η Maude χρησιμοποιεί ως θεμέλιό της την εξισωτική λογική με διατεταγμένους τύπους και συμμετοχές (membership order-sorted equational logic) [23, 7].

Η υπογραφή (signature) μιας εξισωτικής λογικής είναι μια τριάδα (K, Σ, S), όπου:

- Το K είναι ένα σύνολο ειδών (kinds).
- Το $S = \{S_k\}_{k \in K}$ είναι μια οικογένεια ξένων συνόλων τύπων (sorts)· επομένως, κάθε τύπος αντιστοιχεί σε ένα είδος.
- Το $\Sigma = \{\Sigma_{w,k}\}_{(w,k) \in K^* \times K}$ είναι μια οικογένεια συνόλων τελεστών. Κάθε τελεστής έχει μηδέν ή περισσότερα ορίσματα και ένα αποτέλεσμα· κάθε όρισμα, καθώς και το αποτέλεσμα, έχει καθορισμένο είδος.

Η υπογραφή συμβολίζεται και με Σ όταν δεν υπάρχει κίνδυνος παρανόησης.

Οι όροι (terms) ορίζονται αναδρομικά και συγκροτούν το σύνολο $T_{\Sigma,K} = \bigcup_{k \in K} T_{\Sigma,k}$.

- Για κάθε $k \in K$, $\Sigma_k \subseteq T_{\Sigma,k}$.
- Αν $t_i \in T_{\Sigma,k_i} \forall i = 1 \ldots, n$ και $f \in \Sigma_{k_1,\ldots,k_n,k}$, τότε $f(t_1, \ldots, t_n) \in T_{\Sigma,k}$.

Μπορούμε επιπλέον να έχουμε ένα σύνολο μεταβλητών X, όπου κάθε μεταβλητή έχει καθορισμένο είδος. Το σύνολο των όρων που περιέχουν μεταβλητές συμβολίζεται με $T_\Sigma(X)_K$.

Οι ατομικοί τύποι (atomic formulae) μπορούν να είναι δύο ειδών:

- $t = t'$, όπου $t, t' \in T_\Sigma(X)_k$ (Σ-εξισώσεις),
- $t : s$, όπου $t \in T_\Sigma(X)_k$ και $s \in S_k$ (Σ-συμμετοχές),

ενώ οι Σ-προτάσεις (Σ-sentences) ή εξισωτικοί κανόνες (equational rules) έχουν τη μορφή

$$(\forall X)\phi \text{ if } \bigwedge_i p_i = q_i \wedge \bigwedge_j v_j : s_j.$$

όπου το ϕ είναι ένας ατομικός τύπος και όλες οι μεταβλητές των ϕ, p_i, q_i, w_j περιλαμβάνονται στο X.

Ένα ζεύγος αποτελούμενο από μια υπογραφή Σ και ένα σύνολο Σ-προτάσεων E καλείται θεωρία της εξισωτικής λογικής.

Ένα μοντέλο A της εξισωτικής λογικής καλείται Σ-άλγεβρα. Αποτελείται από:

- Ένα σύνολο A_k για κάθε $k \in K$.
- Ένα σύνολο $A_s \subset A_k$ για κάθε $s \in S_k$.

- Μια συνάρτηση $f_A : A_{k_1} \times \cdots \times A_{k_n} \to A_k$ για κάθε $f \in \Sigma_{k_1,\ldots,k_n,k}$.

Τα στοιχεία του A_k που δεν ανήκουν σε κανένα A_s θεωρούνται σφάλματα.

Δεδομένης μιας θεωρίας (Σ, E) της εξισωτικής λογικής, ορίζεται η αρχική άλγεβρα της (initial algebra) $T_{\Sigma/E}$, η οποία είναι ένα μοντέλο της όλα τα στοιχεία του οποίου μπορούν να αναπαρασταθούν από όρους χωρίς μεταβλητές και κάθε δύο όροι που είναι ίσοι μπορούν να αποδειχθούν ίσοι με βάση τους εξισωτικούς κανόνες του E.

2.2.3 Η λογική της αναγραφής

Η λογική της αναγραφής (rewriting logic) είναι γενίκευση της εξισωτικής λογικής που μοντελοποιεί παράλληλους ή μη ντετερμινιστικούς υπολογισμούς. Η Maude μπορεί να θεωρηθεί ως μια υλοποίηση της λογικής της αναγραφής [23, 7].

Μία θεωρία ή προδιαγραφή $\mathcal{R}$ της λογικής της αναγραφής είναι μια πεντάδα (Σ, E, ϕ, L, R), όπου

- το (Σ, E) είναι μια θεωρία της εξισωτικής λογικής
- το $\phi : \Sigma \to \wp(\mathbb{N})$ είναι μια συνάρτηση που για κάθε τελεστή $f \in \Sigma_{k_1,\ldots,k_n}$ επιστρέφει ένα υποσύνολο του $\{1, \ldots, n\}$ που αντιστοιχεί στις θέσεις των ορισμάτων στα οποία δεν μπορούμε να κάνουμε αναγραφές
- το L είναι ένα σύνολο ετικετών (labels)
- το R είναι ένα σύνολο κανόνων αναγραφής (rewriting rules).

Οι ατομικοί τύποι της λογικής της αναγραφής περιλαμβάνουν τους ατομικούς τύπους της υποκείμενης εξισωτικής λογικής, καθώς και τύπους της μορφής $t \to t'$, όπου $t, t' \in T_{\Sigma/E}(X)_k$ για κάποιο $k \in K$.

Οι κανόνες αναγραφής γενικεύουν τους εξισωτικούς κανόνες· έχουν τη μορφή

$$r : (\forall X) t \to t' \text{ if } \bigwedge_i p_i = q_i \wedge \bigwedge_j v_j : s_j \wedge \bigwedge_l w_l \to w'_l,$$

με $r \in L$.

Ένα βασικό και πολύ χρήσιμο χαρακτηριστικό της λογικής της αναγραφής –και επομένως της Maude– είναι ότι έχει ανακλαστικό (reflective) χαρακτήρα, δηλαδή ότι υπάρχει μία «καθολική» πεπερασμένα περιγράψιμη θεωρία αναγραφής $\mathcal{U}$, μέσα στην οποία μπορεί να παρασταθεί οποιαδήποτε πεπερασμένα περιγράψιμη θεωρία αναγραφής (συμπεριλαμβανομένης προφανώς της $\mathcal{U}$).

2.3 Η Maude

Σε αυτή την ενότητα, θα γίνει μια σύντομη παρουσίαση των στοιχείων της Maude που χρησιμοποιούνται στο κεφάλαιο 4 (και στο παράρτημα Α΄)· οι πληροφορίες προέρχονται από τα κεφάλαια 3 έως 6 του [7]. Στην τελευταία υποενότητα, θα γίνει μία νύξη των –πολύ περισσότερων από όσες χρησιμοποιούνται εδώ– δυνατοτήτων της Maude. Τα [7] και [8] προσφέρουν μια αρκετά ικανοποιητική τεκμηρίωση των χαρακτηριστικών και των δυνατοτήτων της.

2.3.1 Βασικά συντακτικά και σημασιολογικά στοιχεία

Το βασικό συντακτικό στοιχείο είναι το αναγνωριστικό (identifier). Ένα αναγνωριστικό είναι μία ειδική περίπτωση συμβολοσειράς χαρακτήρων του ASCII. Το κενό διάστημα, καθώς και οι χαρακτήρες (,), [,], {, } και , χωρίζουν αυτόματα μία συμβολοσειρά σε επιμέρους αναγνωριστικά. Μπορούμε όμως να κατασκευάσουμε αναγνωριστικά που περιλαμβάνουν κάποιον από τους εν λόγω χαρακτήρες, χρησιμοποιώντας τη μορφή μοναδικού αναγνωριστικού (single-identifier form), στην οποία τοποθετούμε το χαρακτήρα ‘ αντί για το χαρακτήρα του κενού και πριν από κάθε άλλο χαρακτήρα που διαχωρίζει τη συμβολοσειρά σε επιμέρους αναγνωριστικά. Για παράδειγμα, η συμβολοσειρά `a b‘c de{f g‘}! f‘(‘)` περιλαμβάνει τα αναγνωριστικά `a`, `b c`, `de`, `{`, `f`, `g}!` και `f()`. Σε πολλές περιπτώσεις, ακόμη κι αν δεν έχουμε χρησιμοποιήσει τη μορφή μοναδικού αναγνωριστικού, η Maude μπορεί να συνάγει ότι μια συμβολοσειρά ορίζει ένα μοναδικό αναγνωριστικό· στον κώδικα που θα παρουσιάσουμε παρακάτω, θα το χρησιμοποιούμε πάντα όταν ορίζουμε (αλλά όχι όταν χρησιμοποιούμε) τελεστές.

Τα αναγνωριστικά συνδυάζονται για να χτίσουν δηλώσεις (declarations) ή ισχυρισμούς (statements). Οι δηλώσεις αφορούν τα είδη των αντικειμένων του προγράμματος και οι ισχυρισμοί τις ιδιότητες που ικανοποιούν τα αντικείμενα. Οι δηλώσεις μπορεί να αφορούν την ύπαρξη τύπων (sorts), οι οποίοι ορίζουν τύπους δεδομένων, την διάταξη των τύπων (subsort relation) ή την ύπαρξη τελεστών (operators) ανάμεσα σε τύπους. Οι ισχυρισμοί περιλαμβάνουν εξισωτικούς κανόνες (equational rules), οι οποίοι δηλώνουν πως δύο αντικείμενα είναι ίσα (με την έννοια της εξισωτικής λογικής), συμμετοχές (memberships), οι οποίες δηλώνουν ότι ένα αντικείμενο ανήκει σε κάποιο τύπο και κανόνες αναγραφής (rewrite rules), οι οποίοι περιγράφουν τοπικές μεταβάσεις σε υποσύνολα καταστάσεων του συστήματος. Κάθε δήλωση ή ισχυρισμός έχει το αναγνωριστικό . στο τέλος του.

Οι δηλώσεις και οι ισχυρισμοί οργανώνονται σε αρθρώματα (modules). Τα αρθρώματα είναι δύο ειδών: συναρτησιακά (functional) και συστήματος (system). Τα πρώτα ξεκινούν με τη λέξη-κλειδί **fmod**, ακολουθούμενη από το όνομα του αρθρώματος και τη λέξη-κλειδί **is**· μπορούν να περιλαμβάνουν δηλώσεις, εξισωτικούς κανόνες και συμμετοχές και τελειώνουν με τη λέξη-κλειδί **endfm**. Αντιστοιχούν σε θεωρίες της εξισωτικής λογικής, αλλά μπορούν να ιδωθούν και ως συναρτησιακά προγράμματα. Τα δεύτερα ξεκινούν με τη λέξη-κλειδί **mod**, ακολουθούμενη από το όνομα του αρθρώματος και τη λέξη-κλειδί **is**· μπορούν να περιλαμβάνουν δηλώσεις και οποιουσδήποτε ισχυρισμούς και τελειώνουν με τη λέξη-κλειδί **endm**. Αντιστοιχούν σε θεωρίες της λογικής της αναγραφής, αλλά μπορούν να ιδωθούν και ως παράλληλα προγράμματα. Τα ονόματα των αρθρωμάτων είθισται να είναι κεφαλαία, με διαφορετικές λέξεις να διαχωρίζονται με -.

Τα σχόλια ξεκινούν με τους χαρακτήρες `---` ή `***` και διαρκούν μέχρι το τέλος της γραμμής.

2.3.2 Δηλώσεις

Τα θεμελιώδη αντικείμενα της εξισωτικής λογικής και της λογικής της αναγραφής είναι οι τύποι, που όπως είπαμε αντιστοιχούν σε τύπους δεδομένων. Για να ορίσουμε έναν τύπο, χρησιμοποιούμε τη λέξι-κλειδί **sort**,

ακολουθούμενη από το αναγνωριστικό (με κάποιους περιορισμούς στους χαρακτήρες που μπορεί να περιλαμβάνει) που θα χρησιμοποιούμε για να συμβολίσουμε τον τύπο. Μπορούμε να ορίσουμε και πολλούς τύπους ταυτόχρονα, χρησιμοποιώντας τη λέξι-κλειδί **sorts**. Κατά σύμβαση, τα ονόματα των τύπων γράφονται σε camel case, δηλαδή κάθε λέξη ξεκινάει με κεφαλαίο και έχει τα υπόλοιπα πεζά και επιμέρους λέξεις παρατίθενται χωρίς κάποιο σύμβολο.

Μία δήλωση της μορφής **subsort** `Sort1 < Sort2 .` δηλώνει πως το `Sort1` είναι υποτύπος του `Sort2`. Η διάταξη των τύπων αντιστοιχεί στη σχέση του περιέχεσθαι στα σύνολα στοιχείων των τύπων και πρέπει να είναι μερική. Όπως και στην εξισωτική λογική, κάθε συνεκτική συνιστώσα τύπων ορίζει ένα είδος (kind). Είναι δυνατή η ύπαρξη όρων που έχουν είδος αλλά όχι τύπο· οι όροι αυτά θεωρούνται σφάλματα. Μπορούμε να αναφερόμαστε σε ένα είδος περικλείοντας έναν οποιοδήποτε τύπο του με τους χαρακτήρες [και].

Μπορούμε να ορίσουμε τελεστές χρησιμοποιώντας τη λέξη-κλειδί **op** (ή **ops** για πολλούς τελεστές ταυτόχρονα), ακολουθούμενη από το σύμβολο :, τους τύπους (αν υπάρχουν) των ορισμάτων, το σύμβολο –>, τον τύπο του αποτελέσματος και, προαιρετικά, μία λίστα από ιδιότητες περικλειόμενη από τους χαρακτήρες [και]. Για παράδειγμα, η δήλωση

ops `s p` : **Int** –> **Int** [**ctor**] .

ορίζει τελεστές `s` και `p` με όρισμα ένα στοιχείο του **Int** και αποτέλεσμα στο **Int**, που έχουν την ιδιότητα **ctor**. Αν ένας τελεστής δεν έχει ορίσματα, τότε καλείται σταθερά.

Τα ονόματα των τελεστών μπορούν να αποτελούνται από ένα ή περισσότερα αναγνωριστικά. Σε περίπτωση που το όνομα ενός τελεστή αποτελείται από πολλά αναγνωριστικά, τότε ο ορισμός του συνίσταται να γίνεται σε μορφή μοναδικού αναγνωριστικού· επιπλέον, αν ο τελεστής περιλαμβάνει κάποια παρένθεση, συνίσταται να εγκλείεται σε παρενθέσεις. Ο χαρακτήρας _ μπορεί να εμφανίζεται στο όνομα ενός τελεστή ακριβώς τόσες φορές όσα είναι τα ορίσματά του. Αν το όνομα του τελεστή δεν περιλαμβάνει το χαρακτήρα _, τότε ο τελεστής χρησιμοποιείται γράφοντας το όνομά του και εγκλείοντας τα ορίσματά του σε παρένθεση, διαχωρισμένα με κόμμα. Αν το όνομα του τελεστή περιλαμβάνει το χαρακτήρα _, τότε μπορεί να χρησιμοποιηθεί και αντικαθιστώντας κάθε εμφάνιση του _ με το αντίστοιχο όρισμα. Τα ονόματα των τελεστών προτείνεται να αποτελούνται –εκτός των συμβόλων– από πεζά γράμματα.

Εκ πρώτης όψεως, τίποτα δεν απαγορεύει σε δύο ή περισσότερους τελεστές να έχουν το ίδιο όνομα, να έχουμε δηλαδή υπερφόρτωση (overloading) ενός τελεστή· υπάρχουν όμως περιπτώσεις που αυτό θα οδηγούσε σε αμφισημία. Επομένως, αν σε κάθε θέση δύο παραλλαγών ενός υπερφορτωμένου τελεστή οι τύποι των ορισμάτων βρίσκονται στο ίδιο είδος (επομένως, η συνωνυμία δύο τελεστών μπορεί όντως να δημιουργήσει περιπτώσεις που δεν ξέρουμε το είδος του αποτελέσματος), τότε τα αποτελέσματά τους πρέπει να βρίσκονται στο ίδιο είδος και επιπλέον (επειδή θα υπάρχουν περιπτώσεις που μπορούν να εφαρμοστούν και οι δύο στα ίδια ορίσματα) οι ιδιότητές τους, με εξαίρεση τις **ctor** και `metadata`, πρέπει να ταυτίζονται.

Από τους τελεστές (και τις μεταβλητές, που θα δούμε αργότερα) χτίζονται επαγωγικά μέσω των τελεστών οι όροι (terms). Λόγω όμως της αυθαίρε-

της σύνταξης των τελεστών, μπορεί ένας όρος να είναι δυνατόν να διαβαστεί με περισσότερους από έναν τρόπους, οδηγούμαστε δηλαδή σε αμφισημία. Για να αποφευχθεί αυτό, κάθε τελεστής συνοδεύεται (από προεπιλογή) από τις ιδιότητες **gather** και **prec**. Η δεύτερη αντιστοιχίζει στον τελεστή έναν φυσικό αριθμό, την προτεραιότητά του· η γενική διαίσθηση είναι ότι τελεστές με χαμηλότερη προτεραιότητα ομαδοποιούνται κατά προτεραιότητα σε έναν όρο. Η πρώτη ορίζει τις δυνατές προτεραιότητες κάθε ορίσματος. Παρακάτω, θα χρειαστεί να αλλάξουμε τις προεπιλεγμένες τιμές μόνο στις ιδιότητες **prec**.

Μερικές ακόμη από τις ιδιότητες που μπορούν να χρησιμοποιηθούν στον ορισμό τελεστών είναι οι **assoc**, **comm**, **id**:, **ctor**, **ditto** και **frozen**. Οι δύο πρώτες δηλώνουν πως ο τελεστής ικανοποιεί την προσεταιριστική και την αντιμεταθετική ιδιότητα αντίστοιχα. Η **id**:, ακολουθούμενη από κάποιο όρο, δηλώνει ότι ο εν λόγω όρος είναι ουδέτερο στοιχείο του τελεστή. Η **ctor** δηλώνει πως το αποτέλεσμα είναι σε κανονική μορφή, δηλαδή δεν μπορεί να απλοποιηθεί περαιτέρω· για παράδειγμα, αν ορίσουμε τους φυσικούς αριθμούς με τα αξιώματα του Peano, τότε το μηδέν και η συνάρτηση του επόμενου κατασκευάζουν όρους σε κανονική μορφή, ενώ ο τελεστής της πρόσθεσης όχι. Η **ditto** χρησιμοποιείται όταν έχουμε υπερφόρτωση τελεστή (στο ίδιο είδος) και αντιγράφει τις ιδιότητες (εκτός της **ctor** και της `metadata`) που ξέρουμε ήδη ότι πρέπει να έχει ο τελεστής. Τέλος, η **frozen**, η οποία μπορεί να πάρει ως ορίσματα τους αύξοντες αριθμούς των ορισμάτων του τελεστή στα οποία αναφέρεται, αλλιώς ισχύει για όλα τα ορίσματα, απαγορεύει τις μεταγραφές[2] σε υποόρους· εδώ θα τη χρησιμοποιήσουμε στον ορισμό των προγραμμάτων του π-λογισμού.

Είναι δυνατόν επίσης να ορίσουμε τελεστές στο επίπεδο των ειδών, κάτι που αντιστοιχεί σε μερικές συναρτήσεις στο επίπεδο των τύπων (προφανώς, σε μια τέτοια περίπτωση πρέπει να καθορίσουμε πότε ο τελεστής έχει αποτέλεσμα· αυτό επιτυγχάνεται με συμμετοχές). Ένας τρόπος που υποστηρίζεται για να το κάνουμε αυτό είναι να αντικαταστήσουμε το σύμβολο `->` με `~>`.

2.3.3 Ισχυρισμοί

Για να κατασκευάσουμε ισχυρισμούς, θα χρειαστούμε μεταβλητές (variables). Κάθε μεταβλητή έχει συγκεκριμένο τύπο και δηλώνεται με τη λέξη-κλειδί **var** ακολουθούμενη από το όνομα της μεταβλητής, το χαρακτήρα : με κενά διαστήματα αριστερά και δεξιά του και τον τύπο (ή το είδος) της· η δήλωση έχει ισχύ για όλο το άρθρωμα. Εναλλακτικά, μπορούμε να ορίσουμε μια μεταβλητή μίας χρήσης (χωρίς τη λέξη-κλειδί **var**, και τα κενά διαστήματα γύρω από το :) στο σημείο που θέλουμε να τη χρησιμοποιήσουμε και με ισχύ που δεν επεκτείνεται πέρα από αυτό. Μπορούμε να δηλώσουμε πολλές μεταβλητές του ίδιου τύπου ταυτόχρονα με τη λέξη-κλειδί **vars**. Κατά σύμβαση, τα ονόματα των μεταβλητών αποτελούνται από κεφαλαίους χαρακτήρες.

Το πρώτο είδος ισχυρισμού που θα περιγράψουμε είναι οι συμμετοχές. Οι συμμετοχές δηλώνουν τον τύπο κάποιου όρου. Κατασκευάζονται με τη

[2] Αυτό σημαίνει πως σε περίπτωση που δεν έχουμε κανόνες αναγραφής, η χρήση της δεν έχει νόημα, παρά μόνο για να υποστηρίξει μελλοντικές επεκτάσεις.

λέξη-κλειδί **mb**, ακολουθούμενη από έναν όρο, το σύμβολο :, έναν τύπο (που πρέπει να ανήκει στο είδος του όρου) και, προαιρετικά, μία λίστα ιδιοτήτων περικλειόμενη από τους χαρακτήρες [και].

Μεγαλύτερο ενδιαφέρον παρουσιάζουν οι εξισωτικοί κανόνες, οι οποίοι δηλώνονται με τη λέξη κλειδί **eq**, ακολουθούμενη από έναν όρο, το σύμβολο = , έναν δεύτερο όρο (στο ίδιο είδος με τον πρώτο) και, προαιρετικά, μία λίστα ιδιοτήτων περικλειόμενη από τους χαρακτήρες [και]. Ο εξισωτικός κανόνας χρησιμοποιείται ως κανόνας αναγωγής για να απλοποιηθεί ο όρος που βρίσκεται στο αριστερό μέλος του και, επομένως, κάθε μεταβλητή που βρίσκεται στο δεξί μέλος πρέπει να εμφανίζεται και στο αριστερό (διαφορετικά, μία αναγωγή θα οδηγεί σε άπειρους δυνατούς όρους). Επιπλέον, η Maude θεωρεί πως κάθε εξισωτικός κανόνας, αν εφαρμοστεί ως κανόνας αναγωγών, τερματίζει και έχει την ιδιότητα Church-Rosser[3], δηλαδή καταλήγει πάντα, ανεξάρτητα από τη σειρά των υπολογισμών, στην ίδια κανονική μορφή. Αυτή η ιδιότητα, η οποία είναι ευθύνη του προγραμματιστή να αποδεικνύεται, συνεπάγεται πως η στρατηγική των απλοποιήσεων που ακολουθεί η Maude δεν αλλοιώνει τη σημασιολογία των θεωριών που ορίζουν τα αρθρώματα.

Στις συμμετοχές και στους εξισωτικούς κανόνες μπορούμε να χρησιμοποιήσουμε την ιδιότητα **label** ακολουθούμενη από ένα αναγνωριστικό, ως ετικέτα που διευκολύνει την αναφορά σε αυτά. Επιπλέον, στους εξισωτικούς κανόνες μπορούμε να χρησιμοποιήσουμε την ιδιότητα **owise**, η οποία δηλώνει πως ο εν λόγω κανόνας πρέπει να χρησιμοποιηθεί μόνο αν οι υπόλοιπες προσπάθειες να απλοποιηθεί ο όρος απέτυχαν. Υπάρχουν και άλλες ιδιότητες ισχυρισμών, αλλά δεν θα μας απασχολήσουν εδώ.

Ακόμη, τόσο οι συμμετοχές όσο και οι εξισωτικοί κανόνες μπορούν να οριστούν ώστε να ισχύουν υπό συνθήκη· αυτό επιτυγχάνεται με την αντικατάσταση των **mb** και **eq** από **cmb** και **ceq** αντίστοιχα και την προσθήκη –πριν από τις ιδιότητες των ισχυρισμών– της λέξης-κλειδί **if** ακολουθούμενης από μία σειρά προϋποθέσεων διαχωρισμένων με τον τελεστή /\. Οι προϋποθέσεις πρέπει να ικανοποιούνται όλες για να εφαρμοστεί ο κανόνας ή να ισχύσει η συμμετοχή, ελέγχονται από αριστερά προς τα δεξιά και μπορούν να είναι όροι είδους [**Bool**] ή εξισωτικοί κανόνες ή συμμετοχές.

Τέλος, πρέπει να αναφέρουμε τους κανόνες αναγραφής: αυτοί έχουν παρόμοια σύνταξη με τους εξισωτικούς κανόνες, με τις διαφορές ότι το **eq** αντικαθίσταται με **rl** και το = με =>. Επιπλέον, οι κανόνες αναγραφής δεν μπορούν να έχουν την ιδιότητα **owise**, μπορούν όμως στις προϋποθέσεις τους να έχουν άλλους κανόνες αναγραφής. Η Maude κάνει τις αναγραφές αφότου έχει ανάγει όλους τους όρους σε κανονική μορφή, επομένως υποθέτει πως το μοντέλο είναι τέτοιο ώστε θα είχε τα ίδια αποτελέσματα αν έκανε αναγραφές πριν τις απλοποιήσεις. Η ιδιότητα αυτή ονομάζεται συνοχή (coherence) και είναι υποχρέωση του προγραμματιστή να την αποδείξει.

[3]Ο τερματισμός και η ιδιότητα Church-Rosser πρέπει να ισχύουν αφότου ληφθούν υπόψιν οι ιδιότητες **assoc** και **comm** των τελεστών, διαφορετικά είναι τετριμμένο να δει κανείς πως ο υπολογισμός δεν μπορεί να τερματίσει. Αυτός είναι ο βαθύτερος λόγος –πέρα από τη συνήθη χρήση τους– που προσφέρονται ως ιδιότητες τελεστών και δεν αφήνονται να υλοποιηθούν ως εξισωτικοί κανόνες.

2.3.4 Ιεραρχίες αρθρωμάτων

Κάθε άρθρωμα αντιστοιχεί σε μια θεωρία της εξισωτικής λογικής ή της λογικής της αναγραφής. Θα θέλαμε να μπορούμε να χρησιμοποιούμε θεωρίες που έχουμε ήδη κατασκευάσει στα πλαίσια ευρύτερων θεωριών. Κάτι τέτοιο είναι εφικτό στην Maude και μπορεί να συμβεί με τρεις τρόπους, για καθέναν από τους οποίους παρέχεται μία λέξη-κλειδί: **protecting**, `extending`, **including**. Αν λοιπόν σε ένα άρθρωμα M_1 γράψουμε την αντίστοιχη λέξη-κλειδί και το όνομα ενός άλλου αρθρώματος, M_2, τότε οι δηλώσεις και οι ισχυρισμοί του M_2 ισχύουν και στο M_1. Η σχέση **protecting** δηλώνει ότι οι κανονικές μορφές των τύπων του M_2 παραμένουν κανονικές και ότι δεν προστίθενται άλλες· η σχέση `extending` δηλώνει ότι οι κανονικές μορφές των τύπων του M_2 παραμένουν κανονικές· η σχέση **including** δεν παρέχει καμία τέτοια εγγύηση. Η Maude δεν ελέγχει την ορθότητα των ισχυρισμών αυτών· η απόδειξή τους αφήνεται στον προγραμματιστή.

2.3.5 Περαιτέρω δυνατότητες της Maude

Όπως αναφέρθηκε νωρίτερα, η λογική της αναγραφής, στην οποία θεμελιώνεται η Maude, έχει ανακλαστικό χαρακτήρα, με την έννοια πως κάθε προδιαγραφή (δηλαδή κάθε άρθρωμα) που ορίζουμε μπορεί να θεωρηθεί ως αντικείμενο μιας ευρύτερης θεωρίας και επομένως να υποστεί επεξεργασία ως τέτοιο. Αυτό δίνει τεράστια ευελιξία στη γλώσσα και έχει οδηγήσει στην υλοποίηση διάφορων χαρακτηριστικών και εργαλείων. Ενδεικτικά, αναφέρουμε τα παρακάτω (τα οποία όμως δεν θα χρησιμοποιηθούν εδώ):

- Ο μη ντετερμινισμός των μοντέλων είναι δυνατόν να οδηγήσει σε περιπτώσεις όπου χρειάζεται να επιλεγεί συγκεκριμένη στρατηγική υπολογισμού που διαφέρει από την προεπιλεγμένη. Κάτι τέτοιο είναι δυνατόν μέσω ανάκλασης και προσφέρεται από την ίδια την Maude.

- Δεδομένου ότι μπορούμε να δούμε τις προδιαγραφές ως δεδομένα, μπορούμε να αποδείξουμε ιδιότητές τους (τερματισμό, Church Rosser, συνοχή), ακόμη και να χρησιμοποιήσουμε την Maude για απόδειξη θεωρημάτων.

- Μπορεί να οριστεί (και έχει οριστεί) μία άλγεβρα αρθρωμάτων, που δίνει τη δυνατότητα για συνδυασμό επιμέρους αρθρωμάτων ή ορισμό γενικευμένων.

- Το πιο σημαντικό επίτευγμα σε αυτή την κατεύθυνση είναι η Full Maude, η οποία είναι μια επέκταση της (Core) Maude που δίνει διάφορες μεταπρογραμματιστικές δυνατότητες και εισάγει μία εναλλακτική object-oriented σύνταξη.

Κεφάλαιο 3

Έλεγχος τύπων προγραμμάτων του π-λογισμού ως προς πολιτικές ιδιωτικότητας

Το παρόν κεφάλαιο αποτελεί επέκταση του [17], το οποίο με τη σειρά του είναι βασισμένο στο [19]. Η επέκταση γίνεται στο πνεύμα του P-RBAC όπως ορίζεται στα [25] και [26]· συνίσταται κυρίως στην εισαγωγή της έννοιας της προϋπόθεσης για την παραχώρηση ενός δικαιώματος και δευτερευόντως στη διχοτόμηση των ομάδων σε χρήστες και ρόλους. Έχει δοθεί προσοχή ώστε η επέκταση να είναι προς τα πίσω συμβατή με το [17], δηλαδή να μπορεί να επεξεργαστεί αντικείμενα που έχουν οριστεί στο μη επεκτεταμένο πλαίσιο και να δώσει την ίδια απάντηση για την ασφάλειά τους.

Η διαδικασία που θα ακολουθήσουμε είναι η εξής: πρώτα θα ορίσουμε μια τυπική γλώσσα περιγραφής πολιτικών ιδιωτικότητας, μετά θα περιγράψουμε το υπολογιστικό μοντέλο του π-λογισμού, έπειτα θα παρουσιάσουμε τον έλεγχο τύπων, ο οποίος ελέγχει αν μια διεργασίας του π-λογισμού είναι σωστά κατασκευασμένη και συνάγει τα δικαιώματα που απαιτούνται για την εκτέλεσή της, και τέλος θα δώσουμε τις αναγκαίες αποδείξεις ορθότητας του τυπικού συστήματος.

Συμβολισμός. Το σύμβολο ~ πάνω από μία μεταβλητή θα σημαίνει ότι η μεταβλητή αυτή αποτελεί σύνολο.

3.1 Η γλώσσα περιγραφής των πολιτικών ιδιωτικότητας

Μια πολιτική ιδιωτικότητας (privacy policy) αποτελεί μία περιγραφή των επιτρεπόμενων (ή μη) ενεργειών που μπορούν να ασκήσουν οι εμπλεκόμενοι στην παραγωγή, επεξεργασία και μεταφορά δεδομένων. Μπορεί να περιγραφεί σε φυσική γλώσσα (πχ νομικά έγγραφα), σε τυπική γλώσσα ή σε συνδυασμό τους.

Στο μοντέλο μας, διάφορες οντότητες, που κατέχουν *ρόλους* (πχ του γιατρού, του υπεύθυνου παραγγελιών) *εξουσιοδοτούνται να ενεργήσουν* (πχ τροποιώντας ή διαβιβάζοντάς τα) πάνω στα *δεδομένα* ενός ή περισσότερων *χρηστών* (πχ του ασθενή, του αγοραστή), για την επίτευξη συγκεκριμένων και ρητών *σκοπών* (πχ τη διάγνωση, την αποστολή παραγγελίας), αν βέβαια συντρέχουν κάποιες *προϋποθέσεις* (πχ η σύμφωνη γνώμη του ασθενή, ηλικία του αγοραστή).

3.1.1 Βασικοί ορισμοί

Ορισμός 3.1. Η γλώσσα των πολιτικών ιδιωτικότητας αποτελείται από τα εξής στοιχεία:

1. Ένα σύνολο βασικών τύπων (Basic Types) $\mathcal{D}$ (τα στοιχεία του θα συμβολίζονται με t), ένα σύνολο σκοπών (Purposes) $\mathcal{PU}$ (τα στοιχεία του θα συμβολίζονται με u), ένα σύνολο χρηστών (Users) $\mathcal{U}$ (τα στοιχεία του θα συμβολίζονται με U), και ένα σύνολο ρόλων (Roles) $\mathcal{R}$ (τα στοιχεία του θα συμβολίζονται με R)· τα σύνολα αυτά θα τα θεωρήσουμε αριθμήσιμα, αλλά προφανώς στην πράξη είναι πεπερασμένα.

2. Τη γλώσσα έκφρασης προϋποθέσεων LC_0, που ορίζεται με τη βοήθεια ενός πεπερασμένου συνόλου μεταβλητών πλαισίου (Context Variables). Κάθε μεταβλητή πλαισίου X διαθέτει ένα πεπερασμένο πεδίο τιμών D_X. Οι προϋποθέσεις (συμβολίζονται με το γράμμα c) ορίζονται επαγωγικά ως εξής:

   - Αν X μία μεταβλητή πλαισίου, $v \in D_X$ και $op \in \{=, \neq\}$, τότε το $(X\ op\ v)$ είναι μια ατομική προϋπόθεση.
   - Αν c_1 και c_2 είναι δύο προϋποθέσεις, τότε η $c_1 \wedge c_2$ είναι προϋπόθεση.

   Αν $X_1, \ldots, X_n$ είναι οι μεταβλητές πλαισίου που εμφανίζονται στην c διατεταγμένες λεξικογραφικά, τότε το σύνολο $D(c) = \prod_{i=1}^{n} D_{X_i}$ καλείται πεδίο τιμών της c. Αν $\vec{v} \in D(c)$, τότε ορίζουμε σχέση $c \vDash \vec{v}$, η οποία διαβάζεται «ικανοποιείται από»:

   - Αν $v \in D_X$, $(X\ =\ v) \vDash v$.
   - Αν $v, u \in D_X$ και $v \neq u$, $(X\ \neq\ v) \vDash u$.
   - Αν $\vec{v}_1 \in D(c_1)$ και $\vec{v}_2 \in D(c_2)$, τότε $\vec{v}_1, \vec{v}_2 \vDash c_1 \wedge c_2 \ \Leftrightarrow\ \vec{v}_1 \vDash c_1 \ \wedge\ \vec{v}_2 \vDash c_2$.

   Μέσω της σχέσης αυτής, το $D(c)$ χωρίζεται σε δύο κλάσεις ισοδυναμίας, τις $D^+(c) = \{\vec{v} \in D(c) \mid c \vDash \vec{v}\}$ (θετικό πεδίο τιμών) και $D^-(c) = \{\vec{v} \in D \mid c \nvDash \vec{v}\}$ (αρνητικό πεδίο τιμών). Κάθε προϋπόθεση ταυτίζεται σημασιολογικά με το θετικό πεδίο τιμών της[1].

3. Το σύνολο των δικαιωμάτων (Permissions) `Perm`, που διακρίνονται σε δύο κατηγορίες: βασικά δικαιώματα, ή αλλιώς δικαιώματα χωρίς προϋποθέσεις, και δικαιώματα με προϋπόθεση (Conditional Permissions).

[1] Είναι εύκολο να δούμε ότι η σχέση $c_1 \sim c_2 \overset{\text{ορ}}{\Leftrightarrow} D^+(c_1) = D^+(c_2)$ είναι σχέση ισοδυναμίας, οπότε κάθε προϋπόθεση αντιπροσωπεύει την κλάση ισοδυναμίας της.

Τα βασικά δικαιώματα που υποστηρίζονται εδώ είναι τα εξής:

`read`	ανάγνωση των δεδομένων
`write`	τροποποίηση των δεδομένων
`access`	πρόσβαση στα δεδομένα
`disc` G	κοινοποίηση των δεδομένων μέσα στην ομάδα G

Ένα δικαίωμα με προϋπόθεση σχηματίζεται από ένα βασικό δικαίωμα, ακολουθούμενο από τη λέξη if και μία προϋπόθεση εκφρασμένη στην LC_0. Τα δικαιώματα θα συμβολίζονται με το γράμμα p.

4. Το σύνολο των ομάδων (Groups) $\mathcal{G} \stackrel{\text{ορ}}{=} \mathcal{U} \cup \mathcal{R}$ (οι ομάδες θα συμβολίζονται με G). Στο σύνολο των ομάδων έχουμε τη μερική διάταξη $\leq_{\mathcal{G}}$, η οποία περιγράφει τον εγκλεισμό ενός ρόλου σε έναν άλλον και στην οποία οι χρήστες παρουσιάζονται μόνο ως ελαχιστικά στοιχεία· δεν θα χρησιμοποιήσουμε όμως την $\leq_{\mathcal{G}}$ απευθείας, αλλά μέσω των ιεραρχιών.

5. Τις ιεραρχίες (Hierarchies), οι οποίες θα συμβολίζονται με το γράμμα H και χτίζονται όπως περιγράφεται από τη γραμματική

$$H ::= \epsilon \quad | \quad U : \tilde{u}[\epsilon] \quad | \quad R : \tilde{u}[\tilde{H}] \ ,$$

όπου το $\tilde{H}$ είναι μη κενό σύνολο ιεραρχιών, το ϵ συμβολίζει την κενή ιεραρχία και $G <_{\mathcal{G}} R \ \forall G \in \bigcup_{H \in \tilde{H}} \text{groups}(H)$· η συνάρτηση $\text{groups}(\cdot)$ ορίζεται επαγωγικά ως εξής:

$$\text{groups}(H) = \begin{cases} \emptyset & \text{αν } H = \epsilon \\ \{G\} \cup \left(\bigcup_{H \in \tilde{H}} \text{groups}(H) \right) & \text{αν } H = G : \tilde{u}[\tilde{H}] \end{cases}$$

Ορίζουμε τη συνάρτηση

$$\text{root}(H) = \begin{cases} \emptyset & \text{αν } H = \epsilon \\ \{G\} & \text{αν } H = G : \tilde{u}[\tilde{H}] \end{cases} \ ,$$

η οποία μας δίνει την ομάδα που βρίσκεται στη ρίζα της ιεραρχίας H, καθώς και τη συνάρτηση

$$u \to H = \begin{cases} \epsilon & \text{αν } H = \epsilon \\ G : (\{u\} \cup \tilde{u})\,[\tilde{H}] & \text{αν } H = G : \tilde{u}[\tilde{H}] \end{cases} \ ,$$

η οποία προσθέτει το σκοπό u στη ρίζα (και επομένως έμμεσα σε όλα τα επίπεδα) της H.

6. Τις συναρτήσεις απόδοσης δικαιωμάτων

$$\pi \ : \ \mathcal{PU} \times \mathcal{G} \to \wp(\texttt{Perm}) \ ,$$

οι οποίες αποδίδουν ένα σύνολο δικαιωμάτων σε μία ομάδα για κάποιο συγκεκριμένο σκοπό.

7. Τις πολιτικές ιδιωτικότητας (συμβολισμός $\mathcal{P}$), που περιγράφονται από την ακόλουθη γραμματική:

$$\mathcal{P} ::= t \gg H, \pi \quad | \quad \mathcal{P}; \mathcal{P}$$

> Η διάταξη των επιμέρους πολιτικών δεν έχει σημασία. Οι πολιτικές που περιλαμβάνουν τον ίδιο βασικό τύπο πάνω από μία φορά θεωρούνται άκυρες· εφεξής, θα ασχοληθούμε μόνο με έγκυρες πολιτικές.

Οι βασικοί τύποι είναι τα δεδομένα των χρηστών· δεν έχουν κάποια δομή, αλλά θα μπορούσαμε να τους δώσουμε ιεραρχική δομή όπως στο Hierarchical P-RBAC [25] ή στο [28]. Οι σκοποί περιγράφουν τους σκοπούς για τους οποίους γίνεται η εκάστοτε ενέργεια πάνω στα δεδομένα· ούτε αυτοί έχουν δομή, αλλά θα μπορούσαν κάλλιστα να αποκτήσουν όπως αναφέρεται στο [25] και αναλυτικά στο [3]. Οι χρήστες αντιπροσωπεύουν τους ίδιους τους χρήστες, ενώ οι ρόλοι αναφέρονται σε θέσεις/ρόλους χρηστών μέσα σε ένα σύστημα (πχ στην αλυσίδα παραγωγής). Τα δικαιώματα περιγράφουν τις δυνατές ενέργειες πάνω στα δεδομένα. Αν κάποια ενέργεια πάνω στα δεδομένα επιτρέπεται μόνο υπό συνθήκη, μοντελοποιείται μέσω των δικαιωμάτων με προϋπόθεση[2]. Οι ιεραρχίες έχουν διπλή χρησιμότητα: κάθε ιεραρχία αφενός περιγράφει ένα υποσύνολο της $\leq_{\mathcal{G}}$ και αφετέρου αποδίδει σε κάθε ομάδα έναν ή περισσότερους σκοπούς για τους οποίους αυτή μπορεί να δουλέψει· η απόδοση ενός σκοπού ισχύει έμμεσα και για τις ομάδες που βρίσκονται χαμηλότερα στην ίδια ιεραρχία. Οι πολιτικές, τέλος, αποτελούν την αναλυτική περιγραφή του ποιες ομάδες έχουν πρόσβαση στα εκάστοτε δεδομένα και –μέσω των συναρτήσεων απόδοσης δικαιωμάτων– ποια ακριβώς δικαιώματα τους δίνονται ανάλογα με το σκοπό.

Συμβολισμός. Μια ιεραρχία της μορφής $G : \tilde{u}[\epsilon]$ θα γράφεται και $G : \tilde{u}$. Μια ιεραρχία της μορφής $G : \emptyset[\tilde{H}]$ με $\tilde{H} \neq \{\epsilon\}$ θα γράφεται και $G[\tilde{H}]$. Μια ιεραρχία της μορφής $G : \emptyset[\epsilon]$ θα γράφεται και $G[\,]$

Σημείωση. Οι συναρτήσεις απόδοσης δικαιωμάτων, και συνακόλουθα οι πολιτικές ιδιωτικότητας, περιγράφουν μόνο την παραχώρηση και όχι την άρνηση δικαιωμάτων. Οποιοδήποτε δικαίωμα δεν παραχωρείται ρητά νοείται ως απαγορευμένο.

3.1.2 Μερικές χρήσιμες σχέσεις διάταξης

Θα ορίσουμε τώρα τη σχέση $\leq$ (θα διαβάζεται «αυστηρότερο από») ανάμεσα στις προϋποθέσεις και ανάμεσα στα δικαιώματα, καθώς και τη σχέση $\lesssim$ (και αυτή θα διαβάζεται «αυστηρότερο από») ανάμεσα στα σύνολα δικαιωμάτων. Πρώτα, όμως, θα χρειαστούμε έναν βοηθητικό ορισμό.

Ορισμός 3.2. Έστω προϋπόθεση $c = \bigwedge_{i=1}^{n} c_i$, όπου c_i ατομικές προϋποθέσεις.

1. Ορίζουμε τη συνάρτηση $\mathrm{vars}(c)$, που μας δίνει το σύνολο των μεταβλητών που εμφανίζονται στην c:

$$\mathrm{vars}(c) = \begin{cases} X & \text{αν } c = (X \ op \ v) \\ \mathrm{vars}(c_1) \cup \mathrm{vars}(c_2) & \text{αν } c = c_1 \wedge c_2 \end{cases}$$

[2] Το πλαίσιο που περιγράφουμε αντιμετωπίζει περιπτώσεις όπου η άσκηση ενός δικαιώματος πάνω στα δεδομένα εξαρτάται μόνο από κάποια εξωτερική συνθήκη και όχι τις περιπτώσεις όπου απαιτείται κάποια ενέργεια (πχ η αποστολή ενός αποδεικτικού)· οι τελευταίες μοντελοποιούνται στο P-RBAC με την έννοια της υποχρέωσης (Obligation) και δεν θα μας απασχολήσουν εδώ.

2. Αν $\text{vars}(c) \cap \tilde{X} \neq \emptyset$, ορίζουμε τον περιορισμό της c στο X (συμβολισμός $c\restriction_{\tilde{X}}$), ως εξής:

$$\left(\bigwedge_{i=1}^{n} c_i\right)\restriction_{\tilde{X}} = \bigwedge_{\tilde{X}\cap\text{vars}(c_i)\neq\emptyset} c_i \ .$$

Προφανώς, $\text{vars}(c\restriction_{\tilde{X}}) \subseteq \tilde{X}$.

Ορισμός 3.3.

1. $c_1 \leq c_2 \overset{\text{ορ}}{\Leftrightarrow} \text{vars}(c_2) \subseteq \text{vars}(c_1) \ \wedge \ D^+(c_1\restriction_{\text{vars}(c_2)}) \subseteq D^+(c_2)$

2. $p_1 \leq p_2 \overset{\text{ορ}}{\Leftrightarrow} p_1 = p_2$
$\vee\ ((p_1 = p \text{ if } c) \ \wedge \ (p_2 = p))$
$\vee\ ((p_1 = p \text{ if } c_1) \ \wedge \ (p_2 = p \text{ if } c_2) \ \wedge \ (c_1 \leq c_2))$

3. $\tilde{p_1} \lesssim \tilde{p_2} \overset{\text{ορ}}{\Leftrightarrow} \forall p \in \tilde{p_1} \exists p' \in \tilde{p_2} : p \leq p'$

Παρατήρηση 3.4. Από τον ορισμό προκύπτει άμεσα ότι $\tilde{p_1} \subseteq \tilde{p_2} \Rightarrow \tilde{p_1} \lesssim \tilde{p_2}$. Στην περίπτωση που τα $\tilde{p_1}$ και $\tilde{p_2}$ δεν περιέχουν δικαιώματα με προϋποθέσεις, παίρνουμε $\tilde{p_1} \lesssim \tilde{p_2} \Leftrightarrow \tilde{p_1} \subseteq \tilde{p_2}$.

Ο ορισμός 3.3 μοντελοποιεί τη διαίσθηση ότι αυστηροποιούμε μία προϋπόθεση αν την κάνουμε να εξαρτάται από περισσότερες μεταβλητές ή αν ελαττώσουμε τα διανύσματα του πεδίου τιμών της που την επαληθεύουν, αυστηροποιούμε ένα δικαίωμα χωρίς προϋπόθεση αν του προσθέσουμε μία προϋπόθεση, αυστηροποιούμε ένα δικαίωμα με προϋπόθεση αν αυστηροποιήσουμε την προϋπόθεση και, τέλος, αυστηροποιούμε ένα σύνολο δικαιωμάτων αν αυστηροποιήσουμε ή αφαιρέσουμε τουλάχιστον ένα δικαίωμα.

Λήμμα 3.5. *Αν* $D^+(c_1) \neq \emptyset$ *και* $\tilde{Y} \subseteq \text{vars}(c_1)$, *τότε*

$$D^+(c_1) \subseteq D^+(c_2) \ \Rightarrow \ D^+(c_1\restriction_{\text{vars}(c_1)\setminus\tilde{Y}}) \subseteq D^+(c_2\restriction_{\text{vars}(c_2)\setminus\tilde{Y}})$$

Απόδειξη. Αν $\tilde{Y} = \emptyset$, ισχύει τετριμμένα. Αλλιώς, έστω $\tilde{X} \overset{\text{ορ}}{=} \text{vars}(c_1)$ (οπότε και $\tilde{X} = \text{vars}(c_2)$). Ονομάζουμε $\tilde{Z} \overset{\text{ορ}}{=} \tilde{X} \setminus \tilde{Y}$ και επειδή $\tilde{X} = \tilde{Y} \cup \tilde{Z}$, $\tilde{Y} \cap \tilde{Z} = \emptyset$, έχουμε $c_1 = c_1\restriction_{\tilde{Y}} \wedge c_1\restriction_{\tilde{Z}}$ και $c_2 = c_2\restriction_{\tilde{Y}} \wedge c_2\restriction_{\tilde{Z}}$.

Έστω τώρα $\vec{v}_{\tilde{Z}} \in D^+(c_1\restriction_{\tilde{Z}})$. Επειδή $D^+(c_1) \neq \emptyset$, μπορούμε να βρούμε $\vec{v}_{\tilde{Y}} \in D^+(c_1\restriction_{\tilde{Y}})$ ώστε $\vec{v} = \vec{v}_{\tilde{Z}}, \vec{v}_{\tilde{Y}} \in D^+(c_1)$. Τότε εξ ορισμού $c_1 \vDash \vec{v}$ και από υπόθεση $c_2 \vDash \vec{v}$. Όμως, από τον ορισμό της $\vDash$ και επειδή $c_2 = c_2\restriction_{\tilde{Y}} \wedge c_2\restriction_{\tilde{Z}}$, $c_2\restriction_{\tilde{Z}} \vDash \vec{v}_{\tilde{Z}}$, δηλαδή $\vec{v}_{\tilde{Z}} \in D^+(c_2\restriction_{\tilde{Z}})$. □

Πρόταση 3.6. *Οι* $\leq$ *του ορισμού 3.3 είναι μερικές διατάξεις, ενώ η* $\lesssim$ *είναι προδιάταξη.*

Απόδειξη.

1.

Ανακλαστική: $c \leq c$, επειδή $c\restriction_{\text{vars}(c)} = c$.

Αντισυμμετρική: Έστω $c_1 \leq c_2$ και $c_2 \leq c_1$. Τότε, $\text{vars}(c_1) = \text{vars}(c_2)$, άρα προφανώς $D^+(c_1\restriction_{\text{vars}(c_2)}) = D^+(c_1)$ και $D^+(c_2\restriction_{\text{vars}(c_1)}) = D^+(c_2)$. Επειδή $c_1 \leq c_2$, παίρνουμε $D^+(c_1) \subseteq D^+(c_2)$ και επειδή $c_2 \leq c_1$, παίρνουμε $D^+(c_2) \subseteq D^+(c_1)$. Άρα τελικά $D^+(c_1) = D^+(c_2)$ και επειδή έχουμε ταυτίσει σημασιολογικά τις προϋποθέσεις με τα αντίστοιχα θετικά πεδία, $c_1 = c_2$

Μεταβατική: Έστω $c_1 \leq c_2$ και $c_2 \leq c_3$. Τότε, έχουμε $D^+(c_1\restriction_{\text{vars}(c_2)}) \subseteq D^+(c_2) = D^+(c_2\restriction_{\text{vars}(c_2)})$, $D^+(c_2\restriction_{\text{vars}(c_3)}) \subseteq D^+(c_3)$ και $\text{vars}(c_3) \subseteq \text{vars}(c_2) \subseteq \text{vars}(c_1)$. Άρα, χρησιμοποιώντας το προηγούμενο λήμμα,

$$\begin{aligned} D^+(c_1\restriction_{\text{vars}(c_3)}) &= D^+(c_1\restriction_{\text{vars}(c_2)\setminus(\text{vars}(c_2)\setminus\text{vars}(c_3))}) \\ &\subseteq D^+(c_2\restriction_{\text{vars}(c_2)\setminus(\text{vars}(c_2)\setminus\text{vars}(c_3))}) \\ &= D^+(c_2\restriction_{\text{vars}(c_3)}) \subseteq D^+(c_3) \end{aligned}$$

2.

Ανακλαστική: Εξ ορισμού.

Αντισυμμετρική: Έστω $p_1 \leq p_2$ και $p_2 \leq p_1$. Τότε, από τον ορισμό, είτε $p_1 = p_2$ και η απόδειξη ολοκληρώνεται είτε $p_1 = p$ if c_1 και $p_2 = p$ if c_2, οπότε έχουμε $c_1 \leq c_2$ και $c_2 \leq c_1$, άρα $c_1 = c_2$ και $p_1 = p_2$.

Μεταβατική: Έστω $p_1 \leq p_2$ και $p_2 \leq p_3$. Προφανώς, αν $p_1 = p_2$ ή $p_2 = p_3$, έχουμε το ζητούμενο. Διαφορετικά, έχουμε είτε $p_1 = p$ if $c_1, p_2 = p$ if $c_2, p_3 = p$ if c_3 με $c_1 \leq c_2 \leq c_3$ είτε $p_1 = p$ if $c_1, p_2 = p$ if $c_2, p_3 = p$ με $c_1 \leq c_2$. Και στις δύο περιπτώσεις, παίρνουμε $p_1 \leq p_3$.

3.

Ανακλαστική: Προφανώς, από την ανακλαστική ιδιότητα των δικαιωμάτων.

Μεταβατική: Έστω $\tilde{p_1} \lesssim \tilde{p_2}$ και $\tilde{p_2} \lesssim \tilde{p_3}$. Τότε, για κάθε $p_1 \in \tilde{p_1}$ υπάρχει $p_2 \in \tilde{p_2}$ ώστε $p_1 \leq p_2$, αλλά για το εν λόγω p_2 υπάρχει $p_3 \in \tilde{p_3}$ ώστε $p_2 \leq p_3$. Άρα, από τη μεταβατική ιδιότητα για τα δικαιώματα, έχουμε το ζητούμενο.

□

Παρατήρηση. Η σχέση $\lesssim$ ανάμεσα σε σύνολα δικαιωμάτων δεν έχει την αντισυμμετρική ιδιότητα. Πράγματι, αν $\tilde{p_1} = \{\texttt{read}, \texttt{read if } c\}$ και $\tilde{p_2} = \{\texttt{read}\}$, έχουμε $\tilde{p_1} \lesssim \tilde{p_2}$, $\tilde{p_2} \lesssim \tilde{p_1}$ και $\tilde{p_1} \neq \tilde{p_2}$.

Σημείωση. Παρότι η LC_0 είναι συντακτικά ίδια με την ομώνυμη γλώσσα που ορίζεται στο Core P-RBAC [25], υπάρχει μία σημαντική διαφορά στη σημασιολογία: στο Core P-RBAC, το αντίστοιχο του $t \gg H, \pi(u, G) = \{p \text{ if } c_1, p \text{ if } c_2\}$ απαιτεί να ικανοποιούνται ταυτόχρονα τα c_1 και c_2 (ως εάν ανάμεσά τους να υπήρχε λογική σύζευξη) για να δοθεί στην ομάδα G το δικαίωμα p στα δεδομένα t για το σκοπό $u\cdot$ όπως όμως προκύπτει από τον ορισμό 3.3 –επειδή έχουμε υπαρκτικό τελεστή στον ορισμό της προδιάταξης των συνόλων δικαιωμάτων– και από τον ορισμό 3.16 που –χρησιμοποιώντας την προδιάταξη των συνόλων δικαιωμάτων– καθορίζει το πότε δίνεται ένα δικαίωμα σε ένα πρόγραμμα του π-λογισμού, στο δικό μας πλαίσιο αρκεί να ικανοποιείται μία από τις δύο συνθήκες (ως εάν ανάμεσά τους να υπήρχε

λογική διάζευξη). Η συμπεριφορά αυτή είναι όμοια με τη συμπεριφορά των κανονικοποιημένων συνόλων ανάθεσης δικαιωμάτων (Normalized Permission Assignment Sets) του Conditional P-RBAC [26]. Όπως σημειώνεται στο [25] και θα δούμε στο επόμενο παράδειγμα, αυτό δημιουργεί κάποιες δυσκολίες στη σωστή διατύπωση προϋποθέσεων.

Παράδειγμα 3.1. Έστω ότι σε μια εταιρεία πωλήσεων έχουμε τις ακόλουθες διατυπώσεις στην πολιτική ιδιωτικότητας:

- Οι υπάλληλοι του τμήματος προώθησης μπορούν να διαβάζουν την ηλεκτρονική διεύθυνση των πελατών για το σκοπό της ηλεκτρονικής προώθησης, εφόσον έχουν πρώτα λάβει τη συγκατάθεση του χρήστη.
- Οι υπάλληλοι του τμήματος προώθησης μπορούν να διαβάζουν την ηλεκτρονική διεύθυνση των ανήλικων χρηστών για το σκοπό της ηλεκτρονικής προώθησης, μόνο εφόσον έχει δώσει τη συγκατάθεσή του ο γονέας.

Μία πρώτη μοντελοποίηση της πολιτικής αυτής θα ήταν

$$\begin{aligned}\text{email} \gg H, \pi(\text{onlinepromotion}, \text{Marketing}) = \\ \{\, &read \text{ if } \text{OwnerConsent} = \text{yes}, \\ &read \text{ if } \text{Age} = \text{under18} \wedge \text{ParentalConsent} = \text{yes} \,\} \quad ,\end{aligned}$$

μοντελοποίηση η οποία δίνει τη δυνατότητα σε μία ανήλικη να παρακάμψει την απαίτηση για συγκατάθεση του γονέα της δίνοντας τη δική της συγκατάθεση. Η σωστή μοντελοποίηση σε αυτή την περίπτωση είναι η ακόλουθη:

$$\begin{aligned}\text{email} \gg H, \pi(\text{onlinepromotion}, \text{Marketing}) = \\ \{\, &\texttt{read} \text{ if } \text{Age} \neq \text{under18} \wedge \text{OwnerConsent} = \text{yes}, \\ &\texttt{read} \text{ if } \text{Age} = \text{under18} \wedge \text{ParentalConsent} = \text{yes} \,\} \quad .\end{aligned}$$

3.1.3 Ένα παράδειγμα πολιτικής ιδιωτικότητας

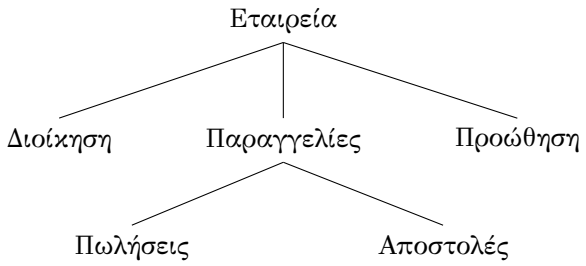
Όπως συμβαίνει και στο [28], θα χρησιμοποιήσουμε την έννοια της υποστασιοποίησης (instantiation), η οποία περιγράφει το τι χρειάζεται ώστε να μπορούμε να κατασκευάσουμε συγκεκριμένες υλοποιήσεις του αφηρημένου πλαισίου που ορίσαμε.

Ορισμός 3.7. Λέμε ότι έχουμε μια υποστασιοποίηση της γλώσσας των πολιτικών ιδιωτικότητας όταν έχουν καθοριστεί πλήρως:

- Ένα σύνολο μεταβλητών πλαισίου $\mathcal{X}$ και για κάθε μεταβλητή $X \in \mathcal{X}$ το πεδίο τιμών της, D_X.
- Ένα σύνολο βασικών τύπων.
- Ένα σύνολο ρόλων, ένα σύνολο χρηστών και μία μερική διάταξη στην ένωσή τους όπως περιγράφεται στον ορισμό 3.1.
- Ένα σύνολο σκοπών.
- Μία πολιτική ιδιωτικότητας με βάση τα παραπάνω σύνολα και το σύνολο των δικαιωμάτων.

Ας περάσουμε τώρα σε ένα παράδειγμα μιας υποστασιοποιημένης πολιτικής ιδιωτικότητας:

Παράδειγμα 3.2. Έστω ότι μελετάμε την πολιτική ιδιωτικότητας μιας εταιρείας ηλεκτρονικών πωλήσεων με τη δομή που φαίνεται στο σχήμα 3.1.



Σχήμα 3.1: Το οργανόγραμμα της εταιρείας του παραδείγματος

Η εταιρεία οφείλει προφανώς να επικοινωνεί με τους πελάτες της για τις πωλήσεις, αλλά επικοινωνεί επίσης με τρίτες εταιρείες, με τις οποίες ανταλλάσσει δεδομένα πελατών της –εφόσον ο πελάτης είναι ενήλικας και έχει δώσει τη συγκατάθεσή του– για διαφημιστικούς σκοπούς. Επιπλέον, το τμήμα διοίκησης κατά καιρούς αναλύει διάφορα στοιχεία των πωλήσεων, όπως τις προτιμήσεις των αγοραστών ανάλογα με την ηλικιακή τους κατηγορία, για να παράξει στατιστικά στοιχεία σχετικά με τη λειτουργία της εταιρείας.

Εδώ θα ασχοληθούμε με τον πελάτη ονόματι Bob και συγκεκριμένα με το πώς χρησιμοποιείται η διεύθυνσή του. Εκτός από τη συνηθισμένη χρήση της, ο Bob έχει δώσει το δικαίωμα στη φίλη του την Alice να χρησιμοποιεί τα στοιχεία του για αγορές από την εταιρεία.

Υποστασιοποιούμε λοιπόν τη γλώσσα των πολιτικών ιδιωτικότητας με τα παρακάτω στοιχεία:

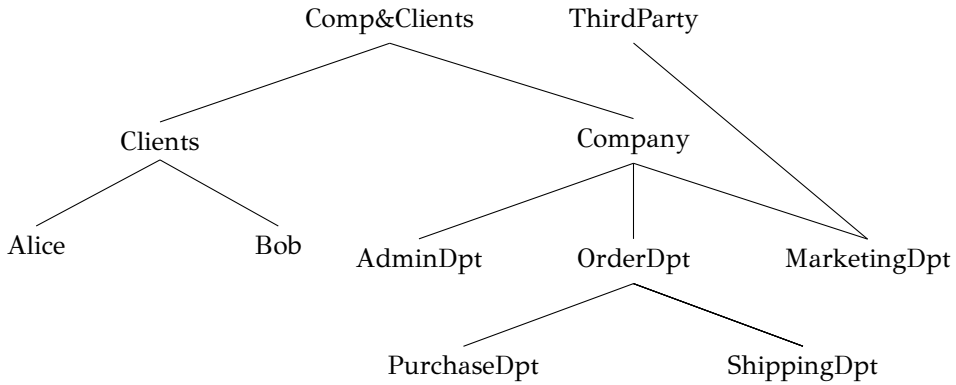
Ως μεταβλητές πλαισίου χρειαζόμαστε μία για την ηλικιακή κατηγορία των χρηστών και μία για τη συγκατάθεσή τους όσον αφορά την προώθηση της διεύθυνσής τους σε τρίτες εταιρείες. Έχουμε λοιπόν

$$\mathcal{X} = \{\text{B.Age}, \text{B.Consent}\}\,,$$
$$D_{\text{Age}} = \{0-17, 18-30, 31-60, \text{over60}\}\,,$$
$$D_{\text{Consent}} = \{\text{Yes}, \text{No}\}\quad.$$

Το ευαίσθητο δεδομένο με το οποίο θα ασχοληθούμε είναι η διεύθυνση, άρα $\mathcal{D} = \{\text{B.Address}\} \cup \mathcal{X}$.

Οι χρήστες που θα μας απασχολήσουν ως τέτοιοι είναι οι $\mathcal{U} = \{\text{Alice}, \text{Bob}\}$, ενώ οι ρόλοι είναι όσοι φαίνονται στο οργανόγραμμα της εταιρείας, ο ρόλος του πελάτη, ένας ρόλος που ενοποιεί τους ρόλους της εταιρείας και του πελάτη και ένας ρόλος που ενοποιεί το ρόλο της τρίτης εταιρείας και του τμήματος προώθησης.

$$\mathcal{R} = \{\,\text{Company}, \text{OrderDpt}, \text{AdminDpt}, \text{PurchaseDpt}, \text{ShippingDpt},$$
$$\text{MarketingDpt}, \text{ThirdParty}, \text{Comp\&Clients}, \text{Clients}\,\}$$



Σχήμα 3.2: Η μερική διάταξη $\leq_{\mathcal{G}}$ των ομάδων που χρησιμοποιούμε για να μοντελοποιήσουμε το σενάριο του παραδείγματος. Δύο ομάδες που συνδέονται με γραμμή σχετίζονται μέσω της $\leq_{\mathcal{G}}$, με την ομάδα που βρίσκεται ψηλότερα στο σχήμα να είναι στο δεξί μέλος της $\leq_{\mathcal{G}}$.

Η μερική διάταξη στο $\mathcal{G} = \mathcal{U} \cup \mathcal{R}$ φαίνεται στο σχήμα 3.2.

Οι σκοποί είναι οι $\mathcal{PU} = \{\text{analysis}, \text{purchase}, \text{marketing}\}$.

Για τον καθορισμό της πολιτικής ιδιωτικότητας, θα δούμε αρχικά τις συναρτήσεις απόδοσης δικαιωμάτων. Έχουμε μία για κάθε στοιχείο του $\mathcal{D}$. Όσον αφορά την ηλικία του Bob, το τμήμα πωλήσεων μπορεί να την αναζητήσει και να τη διαβάσει ώστε να διαπιστώσει αν ο χρήστης έχει την κατάλληλη ηλικία για να προχωρήσει σε αγορά, το τμήμα προώθησης μπορεί να την αναζητήσει και να τη διαβάσει για να διαπιστώσει αν ο πελάτης είναι σε ηλικία που να μπορούν να προωθηθούν τα στοιχεία του και το τμήμα διοίκησης μπορεί να την αναζητήσει και να τη διαβάσει για να τη χρησιμοποιήσει για στατιστική ανάλυση· προφανώς, ο ίδιος ο Bob έχει πλήρη δικαιώματα· συμβολίζουμε $\tilde{p}_{\max} = \{\texttt{read}, \texttt{write}, \texttt{access}, \texttt{disc}\ G \mid G \in \mathcal{G}\}$.

$$\pi_{\text{B.Age}}(u, G) = \begin{cases} \{\texttt{access}, \texttt{read}\} & \text{αν } (u, G) = (\text{analysis}, \text{AdminDpt}) \\ \{\texttt{access}, \texttt{read}\} & \text{αν } (u, G) = (\text{purchase}, \text{PurchaseDpt}) \\ \{\texttt{access}, \texttt{read}\} & \text{αν } (u, G) = (\text{marketing}, \text{MarketingDpt}) \\ \tilde{p}_{\max} & \text{αν } G = \text{Bob} \end{cases}$$

Κατά την πώληση, η διεύθυνση του Bob, εφόσον αυτός είναι ενήλικας, μπορεί να αναζητηθεί από το τμήμα πωλήσεων και να προωθηθεί στο εσωτερικό του τμήματος παραγγελιών, μπορεί να αναζητηθεί και να διαβαστεί από το τμήμα αποστολών, το οποίο όμως δεν υποχρεούται να ελέγξει την ηλικία του Bob, και τέλος μπορεί να χρησιμοποιηθεί -αλλά όχι να τροποποιηθεί- από την Alice. Για το σκοπό της διαφήμισης, το αρμόδιο τμήμα μπορεί, αφότου ελέγξει ότι ο Bob είναι ενήλικας να αναζητήσει τη διεύθυνσή του και, εφόσον έχει δώσει τη συγκατάθεσή του, να την προωθήσει σε τρίτες εταιρείες. Τέλος, ο Bob έχει πλήρη δικαιώματα πάνω στη διεύθυνσή του. Συμβολίζουμε $c_0 = (\text{B.Age} \neq 0 - 17)$, $c_1 = (\text{B.Consent} = \text{Yes})$, $\tilde{p}_{\text{Alice}} =$

$\{\texttt{read}, \texttt{access}, \texttt{disc}\ G \mid G \leq_{\mathcal{G}} \text{Comp\&Clients}\}$.

$$\pi_{\text{B.Address}}(u, G) = \begin{cases} \left\{\begin{array}{l}\texttt{access if } c_0,\\ \texttt{disc OrderDpt if } c_0\end{array}\right\} & \text{αν } u = \text{purchase},\ G = \text{PurchaseDpt} \\ \{\texttt{access}, \texttt{read}\} & \text{αν } u = \text{purchase},\ G = \text{ShippingDpt} \\ \left\{\begin{array}{l}\texttt{access if } c_0,\\ \texttt{disc ThirdParty if } c_1 \wedge c_0\end{array}\right\} & \text{αν } u = \text{marketing},\ G = \text{MarketingDpt} \\ \tilde{p}_{\text{max}} & \text{αν } G = \text{Bob} \\ \tilde{p}_{\text{Alice}} & \text{αν } u = \text{purchase},\ G = \text{Alice} \end{cases}$$

Η συγκατάθεση του Bob για προώθηση των δεδομένων του σε τρίτους για διαφημιστικούς σκοπούς αφορά κυρίως το αντίστοιχο τμήμα, που μπορεί να την αναζητήσει και να τη διαβάσει· αφορά επίσης το τμήμα διοίκησης για το σκοπό της στατιστικής ανάλυσης, που μπορεί επίσης να την αναζητήσει και να τη διαβάσει· προφανώς, και πάλι ο Bob έχει πλήρη δικαιώματα.

$$\pi_{\text{B.Consent}}(u, G) = \begin{cases} \{\texttt{access}, \texttt{read}\} & \text{αν } (u, G) = (\text{marketing}, \text{MarketingDpt}) \\ \{\texttt{access}, \texttt{read}\} & \text{αν } (u, G) = (\text{analysis}, \text{AdminDpt}) \\ \tilde{p}_{\text{max}} & \text{αν } G = \text{Bob} \end{cases}$$

Χρησιμοποιούμε ως ιεραρχία το υποδέντρο της σχέσης $\leq_{\mathcal{G}}$ με ρίζα το Comp&Clients.

$H =$ Comp&Clients [
  Clients : {purchase} [Alice[ ], Bob[ ]],
  Company [
    AdminDpt : {analysis} ,
    OrderDpt : {purchase} [PurchaseDpt[ ], ShippingDpt[ ]],
    MarketingDpt : {marketing} ,
  ],
]

Έχουμε λοιπόν ότι η πολιτική ιδιωτικότητας του παραδείγματος είναι η

$\mathcal{P}_{\text{Bob}} = \text{B.Age} \gg H, \pi_{\text{B.Age}}; \text{B.Address} \gg H, \pi_{\text{B.Address}}; \text{B.Consent} \gg H, \pi_{\text{B.Consent}}$

3.2 Ο π-λογισμός

Ο π-λογισμός είναι ένα Turing-complete υπολογιστικό μοντέλο το οποίο περιγράφει παράλληλους υπολογισμούς μέσω διαδικασιών που τρέχουν παράλληλα, δημιουργούν κανάλια και τα χρησιμοποιούν για να ανταλλάξουν μηνύματα· τα μηνύματα αυτά μπορούν να είναι και ονόματα καναλιών. Έτσι, τα συστήματα του π-λογισμού έχουν ενός είδους δυναμικό χαρακτήρα. Υπάρχουν πολλές παραλλαγές του π-λογισμού· αυτή που θα χρησιμοποιήσουμε εδώ είναι αυτή που περιγράφεται στο [4] και χρησιμοποιείται στο [17], επεκτεταμένη ώστε να υποστηρίζεται ο έλεγχος προϋποθέσεων.

3.2.1 Συντακτικό

Ορισμός 3.8. Τα στοιχεία του π-λογισμού ορίζονται ως εξής:

1. Έχουμε ως βασικά στοιχεία ένα σύνολο σκοπών, το οποίο ταυτίζουμε με το σύνολο $\mathcal{PU}$ του ορισμού 3.8, ένα σύνολο ομάδων, το οποίο ταυτίζουμε με το σύνολο $\mathcal{G}$ του ορισμού 3.8 και ένα αριθμήσιμο σύνολο ονομάτων (Names) $\mathcal{N}$, τα στοιχεία του οποίου θα συμβολίζουμε με x, y, z, a, b. Κάθε σύνολο τιμών μεταβλητής πλαισίου το συμπεριλαμβάνουμε ως υποσύνολο του $\mathcal{N}$.

2. Έχουμε επιπλέον το σύνολο των τύπων (Types) $\mathcal{T}$ (τα στοιχεία του θα συμβολίζονται με T). Στην εφαρμογή μας, το $\mathcal{T}$ περιλαμβάνει

   - κάθε βασικό τύπο $t \in \mathcal{D}$.
   - κάθε μεταβλητή πλαισίου X.
   - κάθε στοιχείο της μορφής $G[T]$, όπου T τύπος.

3. Τα προγράμματα του π-λογισμού ορίζονται εδώ σε δύο επίπεδα: τις διεργασίες και τα συστήματα. Οι διεργασίες (Processes), που θα συμβολίζονται με P, ορίζονται με βάση την ακόλουθη γραμματική:

$$P ::= \mathbf{0} \mid P1 \mid P2 \mid !P \mid (\nu\, x : T)P \mid x(y : T).P \\ \mid \bar{x}\,\langle y\rangle .P \mid [x = v](P1; P2) \mid [\![x = v]\!]P \mid [\![x \neq v]\!]P$$

   Στις διεργασίες της μορφής $x(y : T).P$ και $(\nu\, y : T)P$, το y δεν μπορεί να είναι τιμή μεταβλητής πλαισίου. Αντιθέτως, στις διεργασίες της μορφής $[x = v](P1; P2)$, $[\![x = v]\!]P$ και $[\![x \neq v]\!]P$, το v οφείλει να είναι τιμή μεταβλητής πλαισίου.
   Τα συστήματα (Systems), που θα συμβολίζονται με S, ορίζονται με βάση την ακόλουθη γραμματική:

$$S ::= \mathbf{0} \mid S1 \mid S2 \mid (\nu\, x : T)S \mid (\nu\, R)S \mid (\nu\, G)P\,\langle u\rangle$$

Κάθε όνομα έχει έναν τύπο. Ένας τύπος της μορφής $G[T]$ δηλώνει ότι το όνομα στο οποίο αντιστοιχεί είναι ένα κανάλι που μπορεί να χρησιμοποιηθεί στα πλαίσια της ομάδας G για την αποστολή ή τη λήψη μηνυμάτων τύπου T· αν το t είναι βασικός τύπος, τότε τύποι της μορφής $G[t]$ δηλώνουν ότι διαβάζεται ή γράφεται η τιμή του t.

Τα προγράμματα $\mathbf{0}_S$ και $\mathbf{0}_P$ είναι το κενό σύστημα και η κενή διεργασία αντίστοιχα. Ο τελεστής $\mid$ σημαίνει ότι δύο διεργασίες ή δύο συστήματα τρέχουν παράλληλα. Το $!P$ σημαίνει ότι η διεργασία P τρέχει δυνάμει άπειρες φορές. Με το $(\nu\, x : T)$, δεσμεύεται το όνομα x με τύπο T στη διεργασία ή το σύστημα που ακολουθεί· το y δεν μπορεί να είναι τιμή μεταβλητής πλαισίου, διότι αυτές παίζουν το ρόλο οικουμενικών σταθερών στο σύστημά μας και επομένως δεν έχει νόημα η δέσμευσή τους σε συγκεκριμένο περιβάλλον. Η διεργασία $\bar{x}\,\langle y\rangle .P$ χρησιμοποιεί το κανάλι x για να στείλει το μήνυμα y και μετά συνεχίζει ως P. Η διεργασία $x(y : T).P$ χρησιμοποιεί το κανάλι x για να λάβει ένα μήνυμα y τύπου T, δεσμεύοντας ταυτόχρονα το όνομα y, και μετά συνεχίζει ως P· το y δεν μπορεί να είναι τιμή μεταβλητής πλαισίου, διότι αυτές παίζουν το ρόλο οικουμενικών σταθερών στο σύστημά μας και

επομένως είναι a priori γνωστές (δεν έχει νόημα η ρητή ανάγνωση της τιμής τους). Η διεργασία $[x = v](P_1; P_2)$ ελέγχει αν το όνομα x ταυτίζεται με το όνομα v· αν ναι, συνεχίζει ως P_1, ενώ αν όχι, συνεχίζει ως P_2. Η διεργασία $[\![x = y]\!]P$ θεωρεί δεδομένο ότι $x = y$ και συνεχίζει ως P, ενώ η διεργασία $[\![x \neq y]\!]P$ θεωρεί δεδομένο ότι $x \neq y$ και συνεχίζει ως P· οι δύο αυτές συντακτικές κατασκευές λειτουργούν ως «ετικέτες» που δηλώνουν πως συνέβη ένας έλεγχος προϋπόθεσης, πληροφορία που αλλιώς θα ξεχνιόταν αλλά στο πλαίσιό μας έχει μεγάλη σημασία και χρησιμοποιείται εξάλλου στον έλεγχο τύπων. Σε ενδεχόμενη εφαρμογή, οι $[\![x = y]\!]P$ και $[\![x \neq y]\!]P$ δεν χρειάζεται –και αποθαρρύνεται– να χρησιμοποιούνται από τον χρήστη. Το $(\nu\, G)P\,\langle u\rangle$ δεσμεύει το G και ορίζει ότι μία διεργασία τρέχει εκ μέρους της ομάδας (κάποιου ρόλου ή κάποιου χρήστη) G για το σκοπό u. Τέλος, με το $(\nu\, R)S$, ο ρόλος R δεσμεύεται στο σύστημα S και δηλώνεται ότι το S έχει και τα δικαιώματα του R. Θεωρούμε ότι κάθε σύστημα τρέχει μόνο εκ μέρους των ομάδων που αναφέρονται ρητά σε αυτό.

Συμβολισμός.

1. Για συντομία, $[x = y]P \overset{\text{ορ}}{\equiv} [x = y](P; \mathbf{0})$ και $[x \neq y]P \overset{\text{ορ}}{\equiv} [x = y](\mathbf{0}; P)$.

2. Τα ελεύθερα ονόματα σε μία διεργασία ή ένα σύστημα θα συμβολίζονται $\mathrm{fn}(P)$ και $\mathrm{fn}(S)$ αντίστοιχα, ενώ τα δεσμευμένα θα συμβολίζονται $\mathrm{bn}(P)$ και $\mathrm{bn}(S)$ αντίστοιχα. Οι ελεύθερες ομάδες σε έναν τύπο, μια διεργασία ή ένα σύστημα θα συμβολίζονται $\mathrm{fg}(T)$, $\mathrm{fg}(P)$ και $\mathrm{fg}(S)$ αντίστοιχα. Οι δεσμευμένες ομάδες σε ένα σύστημα θα συμβολίζονται $\mathrm{bg}(S)$. Παρατηρούμε ότι, με βάση τους ορισμούς που έχουμε δώσει, οι

T_0	$\mathrm{fg}(T_0)$
t	$\emptyset$
$G[T]$	$\{G\} \cup \mathrm{fg}(T)$

P_0	$\mathrm{fn}(P_0)$	$\mathrm{bn}(P_0)$	$\mathrm{fg}(P_0)$
$\mathbf{0}$	$\emptyset$	$\emptyset$	$\emptyset$
$P1 \mid P2$	$\mathrm{fn}(P_1) \cup \mathrm{fn}(P_2)$	$\mathrm{bn}(P_1) \cup \mathrm{bn}(P_2)$	$\mathrm{fg}(P_1) \cup \mathrm{fg}(P_2)$
$!P$	$\mathrm{fn}(P)$	$\mathrm{bn}(P)$	$\mathrm{fg}(P)$
$(\nu\, x : T)P$	$\mathrm{fn}(P) \setminus \{x\}$	$\{x\} \cup \mathrm{bn}(P)$	$\mathrm{fg}(T) \cup \mathrm{fg}(P)$
$x(y : T).P$	$\{x\} \cup \mathrm{fn}(P) \setminus \{y\}$	$\{y\} \cup \mathrm{bn}(P)$	$\mathrm{fg}(T) \cup \mathrm{fg}(P)$
$\overline{x}\,\langle y\rangle\,.P$	$\{x, y\} \cup \mathrm{fn}(P)$	$\mathrm{bn}(P)$	$\mathrm{fg}(P)$
$[x = v](P1; P2)$	$\{x, v\} \cup \mathrm{fn}(P_1) \cup \mathrm{fn}(P_2)$	$\mathrm{bn}(P_1) \cup \mathrm{bn}(P_2)$	$\mathrm{fg}(P_1) \cup \mathrm{fg}(P_2)$
$[\![x\ op\ v]\!]P$	$\{x, v\} \cup \mathrm{fn}(P)$	$\mathrm{bn}(P)$	$\mathrm{fg}(P)$

S_0	$\mathrm{fn}(S_0)$	$\mathrm{bn}(S_0)$	$\mathrm{fg}(S_0)$	$\mathrm{bg}(S_0)$
$\mathbf{0}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$S1 \mid S2$	$\mathrm{fn}(S_1) \cup \mathrm{fn}(S_2)$	$\mathrm{bn}(S_1) \cup \mathrm{bn}(S_2)$	$\mathrm{fg}(S_1) \cup \mathrm{fg}(S_2)$	$\mathrm{bg}(S_1) \cup \mathrm{bg}(S_2)$
$(\nu\, x : T)S$	$\mathrm{fn}(S) \setminus \{x\}$	$\{x\} \cup \mathrm{bn}(S)$	$\mathrm{fg}(T) \cup \mathrm{fg}(S)$	$\mathrm{bg}(S)$
$(\nu\, R)S$	$\mathrm{fn}(S)$	$\mathrm{bn}(S)$	$\mathrm{fg}(S) \setminus \{R\}$	$\{R\} \cup \mathrm{bg}(S)$
$(\nu\, G)P\,\langle u\rangle$	$\mathrm{fn}(P)$	$\mathrm{bn}(P)$	$\mathrm{fg}(P) \setminus \{G\}$	$\{G\}$

Σχήμα 3.3: Πίνακες που συνοψίζουν τα περί ελεύθερων και δεσμευμένων ομάδων και ονομάτων σε διεργασίες, συστήματα και τύπους.

τιμές μεταβλητών πλαισίου δεν εμφανίζονται ποτέ δεσμευμένες.

Ορισμός 3.9. Το σύμβολο $P\{x/y\}$ (αντ. $S\{x/y\}$) θα δηλώνει ότι οι ελεύθερες εμφανίσεις του y στην διεργασία P (αντ. στο σύστημα S) αντικαθίστανται ομοιόμορφα με το x· αν το x ήδη εμφανίζεται ως δεσμευμένο όνομα στην P (αντ. στο S), πρέπει πρώτα να μετονομαστεί –η μετονομασία αυτή σχετίζεται με την έννοια της α-ισοδυναμίας που θα δούμε στην επόμενη υποενότητα.

3.2.2 Σημασιολογία

Η σημασιολογία των προγραμμάτων του π-λογισμού ορίζεται με πολλούς τρόπους, ανάλογα με τις επιθυμητές ιδιότητες (ενδεικτικά κάποιες συνηθισμένες παραλλαγές στο [29]).

Συμβολισμός. Για την εν λόγω υποενότητα, για συντομία θα χρησιμοποιούμε το γράμμα F για να αναφερθούμε σε αντικείμενα που μπορούν να είναι είτε διεργασίες είτε συστήματα.

Κατ' αρχάς, παρατηρούμε ότι δύο συντακτικά διαφορετικές διεργασίες (ή συστήματα) μπορούν να έχουν την ίδια συμπεριφορά.

Το ένα είδος τέτοιων περιπτώσεων έχει να κάνει με την επιλογή των δεσμευμένων ονομάτων, όπως παραδείγματος χάριν οι $P_1 = (\nu\, x : T_1)x(y : T_2).\bar{y}\langle z\rangle.\mathbf{0}$ και $P_2 = (\nu\, a : T_1)a(b : T_2).\bar{b}\langle z\rangle.\mathbf{0}$, οι οποίες στέλνουν και οι δύο το z μέσα από ένα κανάλι (ίδιου τύπου T_1) του οποίου το όνομα δέχονται μέσω ενός τρίτου καναλιού (ίδιου τύπου T_2). Η κατάσταση θυμίζει την επιλογή μεταβλητής σε έναν ποσοδείκτη ενός τύπου της κατηγορηματικής λογικής ή την επιλογή μεταβλητής ολοκλήρωσης. Όπως και εκεί, θέλουμε να μη γίνεται σημασιολογική διάκριση· χρησιμοποιούμε για αυτό την έννοια της α-ισοδυναμίας (a-equivalence).

Ορισμός 3.10. Έστω διεργασίες ή συστήματα F_1, F_2. Τα F_1, F_2 λέγονται α-ισοδύναμα (συμβολισμός $F_1 \equiv_\alpha F_2$) αν ισχύει κάτι από τα παρακάτω:

- $F_1 = F_2$.
- $F_1 = !P$, $F_2 = !P'$ και $P \equiv_\alpha P'$.
- $F_1 = x(y : T).P$, $F_2 = x(z : T).P'$ και υπάρχει $b \notin \mathrm{fn}(P) \cup \mathrm{bn}(P) \cup \mathrm{fn}(P') \cup \mathrm{bn}(P')$ ώστε $P\{b/y\} \equiv_\alpha P'\{b/z\}$.
- $F_1 = x\langle y\rangle.P$, $F_2 = x\langle y\rangle.P'$ και $P \equiv_\alpha P'$.
- $F_1 = [x = y](P_1; P_2)$, $F_2 = [x = y](P_1'; P_2')$ και ισχύουν $P_1 \equiv_\alpha P_1'$ και $P_2 \equiv_\alpha P_1'$.
- $F_1 = [\![x\ op\ y]\!]P$, $F_2 = [\![x\ op\ y]\!]P'$ και $P \equiv_\alpha P'$.
- $F_1 = (\nu\, x : T)F$, $F_2 = (\nu\, y : T)F'$ και υπάρχει $z \notin \mathrm{fn}(F) \cup \mathrm{bn}(F) \cup \mathrm{fn}(F') \cup \mathrm{bn}(F')$ ώστε $F\{z/x\} \equiv_\alpha F'\{z/y\}$.
- $F_1 = F_1'' \mid F_2''$, $F_2 = F_1' \mid F_2'$ και ισχύουν $F_1'' \equiv_\alpha F_1'$ και $F_2'' \equiv_\alpha F_2'$.
- $F_1 = (\nu\, G)P\langle u\rangle$, $F_2 = (\nu\, G)P'\langle u\rangle$ και $P \equiv_\alpha P'$.
- $F_1 = (\nu\, G)S$, $F_2 = (\nu\, G)S'$ και $S \equiv_\alpha S'$.

Συνέπεια. *Δύο διεργασίες ή συστήματα είναι α-ισοδύναμα αν διαφέρουν μόνο στην επιλογή των δεσμευμένων ονομάτων τους.*

Συνέπεια. *Η α-ισοδυναμία είναι σχέση ισοδυναμίας.*

Οι υπόλοιπες περιπτώσεις που δύο συντακτικά διαφορετικές διεργασίες (ή συστήματα) θέλουμε να ταυτιστούν σημασιολογικά μπορούν να αντιμετωπιστούν –αν και κάτι τέτοιο δεν είναι αναγκαίο για το σκοπό μας– με την έννοια της ομοιότητας (congruence)· τέτοιες παραδείγματος χάριν είναι οι περιπτώσεις που δεν μας ενδιαφέρει η σειρά που παρουσιάζονται οι επιμέρους διεργασίες σε μία μεγαλύτερη (πχ οι $P_1 \mid P_2$ και $P_2 \mid P_1$ έχουν και οι δύο διαισθητικά την ίδια συμπεριφορά). Ανάλογα με την εφαρμογή, οι κανόνες βάσει των οποίων δύο διεργασίες θεωρούνται όμοιες μπορεί να διαφέρουν. Εδώ μπορούμε να χρησιμοποιήσουμε τον ακόλουθο ορισμό:

Ορισμός 3.11. Η σχέση ομοιότητας $\equiv$ ανάμεσα σε διεργασίες και ανάμεσα σε συστήματα είναι η ελάχιστη σχέση ισοδυναμίας για την οποία

1. $F_1 \equiv_\alpha F_2 \;\Rightarrow\; F_1 \equiv F_2$
2. $F_1 \mid F_2 \equiv F_2 \mid F_1$, $(F_1 \mid F_2) \mid F_3 \equiv F_1 \mid (F_2 \mid F_3)$, $F \mid \mathbf{0} \equiv F$
3. $x \neq y \;\Rightarrow\; (\nu\, x : T_1)(\nu\, y : T_2)F \equiv (\nu\, y : T_2)(\nu\, x : T_1)F$
4. $x \notin \mathrm{fn}(F_1) \;\Rightarrow\; (\nu\, x : T)(F_1 \mid F_2) \equiv F_1 \mid (\nu\, x : T)F_2$
5. $x \notin \{y, z\} \;\Rightarrow\; (\nu\, x : T)[y == z](P_1; P_2) \equiv [y == z]((\nu\, x : T)P_1; (\nu\, x : T)P_2)$
6. $x \notin \{y, z\} \;\Rightarrow\; (\nu\, x : T)[[y \; op \; z]]P \equiv [[y \; op \; z]](\nu\, x : T)P$
7. $!P \equiv !P \mid P$
8. $G \notin \mathrm{fg}(T) \;\Rightarrow\; (\nu\, G)(\nu\, x : T)S \equiv (\nu\, x : T)(\nu\, G)S$
9. $(\nu\, G)(S_1 \mid S_2) \equiv (\nu\, G)S_1 \mid (\nu\, G)S_2$
10. Η $\equiv$ σέβεται τη δομή των συστημάτων και των διεργασιών, δηλαδή
    - $P_1 \equiv P_2 \;\Rightarrow\; !P_1 \equiv !P_2$
    - $P_1 \equiv P_2 \;\Rightarrow\; x(y : T)P_1 \equiv x(y : T)P_2$
    - $P_1 \equiv P_2 \;\Rightarrow\; \bar{x}\,\langle y \rangle\, P_1 \equiv \bar{x}\,\langle y \rangle\, P_2$
    - $P_1 \equiv P_2 \;\Rightarrow\; [[x \; op \; y]]P_1 \equiv [[x \; op \; y]]P_2$
    - $P_1 \equiv P_1' \;\wedge\; P_2 \equiv P_2' \;\Rightarrow\; [x == y](P_1; P_2) \equiv [x == y](P_1'; P_2')$
    - $F_1 \equiv F_1' \;\Rightarrow\; F_1 \mid F_2 \equiv F_1' \mid F_2$
    - $F_1 \equiv F_2 \;\Rightarrow\; (\nu\, x : T)F_1 \equiv (\nu\, x : T)F_2$
    - $P_1 \equiv P_2 \;\Rightarrow\; (\nu\, G)P_1\,\langle u \rangle \equiv (\nu\, G)P_2\,\langle u \rangle$
    - $S_1 \equiv S_2 \;\Rightarrow\; (\nu\, G)S_1 \equiv (\nu\, G)S_2$
11. $x \notin \mathrm{fn}(F) \;\Rightarrow\; (\nu\, x : T)F \equiv F$

Το σημείο 1 ορίζει την α-ισοδυναμία ως ειδική περίπτωση της ομοιότητας. Το σημείο 2 δηλώνει πως η σύνθεση επιμέρους διεργασιών ή συστημάτων που τρέχουν παράλληλα έχει τη δομή αντιμεταθετικού μονοειδούς. Τα σημεία 3, 5, 6, 8 και 9 καθορίζουν πότε μία δέσμευση ονόματος μπορεί να αντιμετατεθεί με ένα άλλο συντακτικό στοιχείο. Το σημείο 4 είναι ένα από τα σημαντικότερα, διότι δείχνει ότι διαφορετικές διεργασίες μπορούν να καταλήξουν να μοιράζονται ένα όνομα και δίνει στον π-λογισμό μεγάλη εκφραστικότητα –αλλά και πολυπλοκότητα. Το σημείο 7 μοντελοποιεί τη διαίσθησή μας για τη σημασία των «δυνάμει άπειρων επαναλήψεων». Το σημείο 10 μετατρέπει τη σχέση ισοδυναμίας σε σχέση ομοιότητας. Τέλος, το σημείο 11 χρησιμεύει ως «garbage collection», απομακρύνοντας δεσμευμένα ονόματα που δεν χρησιμοποιούνται· από αυτό συνεπάγεται ότι $(\nu\, x : T)\mathbf{0} \equiv \mathbf{0}$.

Το βασικό χαρακτηριστικό του π-λογισμού είναι ότι οι διεργασίες του επικοινωνούν ανταλλάσσοντας μηνύματα. Αυτό λοιπόν είναι και το κεντρικό ζήτημα της σημασιολογίας. Οι πιο συχνά χρησιμοποιούμενοι ορισμοί είναι η σημασιολογία με αναγωγές (reduction semantics) και η σημασιολογία των μεταβάσεων με ετικέτες (labelled transition semantics)· εδώ θα χρησιμοποιήσουμε τη σημασιολογία των μεταβάσεων με ετικέτες, επειδή, όπως εξηγείται στο [19], η σημασιολογία με αναγωγές θα συνδυαζόταν δύσκολα με την επιθυμητή σημασία της δέσμευσης μιας ομάδας σε ένα σύστημα.

Ορισμός 3.12.

1. Οι ετικέτες ορίζονται με βάση την ακόλουθη γραμματική:

$$l \;::=\; \tau \quad | \quad x(y:T) \quad | \quad \bar{x}\,\langle y\rangle \quad | \quad (\nu\, y:T)\bar{x}\,\langle y\rangle \;\;,$$

όπου η τ συμβολίζει την εσωτερική (internal) ή σιωπηλή (silent) μετάβαση, η $x(y:T)$ συμβολίζει τη μετάβαση όπου η διεργασία δέχεται ένα όνομα, η $\bar{x}\,\langle y\rangle$ συμβολίζει τη μετάβαση όπου η διεργασία στέλνει ένα όνομα και η $(\nu\, y:T)\bar{x}\,\langle y\rangle$ συμβολίζει τη μετάβαση όπου η διεργασία στέλνει ένα δεσμευμένο όνομα. Οι συναρτήσεις $\mathrm{fn}(l)$ και $\mathrm{bn}(l)$ δίνουν τα ελεύθερα και τα δεσμευμένα ονόματα μίας ετικέτας αντίστοιχα και η συνάρτηση $\mathrm{tn}(l)$ δίνει τα ονόματα των οποίων ο τύπος συμπεριλαμβάνεται στην ετικέτα, σύμφωνα με τον ακόλουθο πίνακα

l	$\mathrm{fn}(l)$	$\mathrm{bn}(l)$	$\mathrm{tn}(l)$
τ	$\emptyset$	$\emptyset$	$\emptyset$
$x(y:T)$	$\{x,y\}$	$\emptyset$	$\{y\}$
$\bar{x}\,\langle y\rangle$	$\{x,y\}$	$\emptyset$	$\emptyset$
$(\nu\, y:T)\bar{x}\,\langle y\rangle$	$\{x\}$	$\{y\}$	$\{y\}$

Ορίζουμε επίσης τη σχέση $\mathrm{dual}(\cdot,\cdot)$ ανάμεσα σε ετικέτες, ως εξής:

$$\mathrm{dual}(l_1,l_2) \overset{\text{ορ}}{\Leftrightarrow} \{l_1,l_2\} = \{x(y:T), \bar{x}\,\langle y\rangle\} \vee \{l_1,l_2\} = \{x(y:T), (\nu\, y:T)\bar{x}\,\langle y\rangle\} \;\;.$$

2. Η σημασιολογία των μεταβάσεων με ετικέτες είναι η ακόλουθη

$$x(y:T).P \xrightarrow{x(z:T)} P\{z/y\} \qquad \text{(In)}$$

$$\bar{x}\,\langle y\rangle\,.P \xrightarrow{\bar{x}\langle y\rangle} P \qquad \text{(Out)}$$

$$\frac{F_1 \xrightarrow{l} F_1' \qquad \mathrm{bn}(l) \cap \mathrm{fn}(F_2) = \emptyset}{F_1 \mid F_2 \xrightarrow{l} F_1' \mid F_2} \qquad \text{(ParL)}$$

$$\frac{F_2 \xrightarrow{l} F_2' \qquad \mathrm{bn}(l) \cap \mathrm{fn}(F_1) = \emptyset}{F_1 \mid F_2 \xrightarrow{l} F_1 \mid F_2'} \quad \text{(ParR)}$$

$$\frac{F \xrightarrow{l} F' \qquad x \notin \mathrm{fn}(l)}{(\nu\, x : T)F \xrightarrow{l} (\nu\, x : T)F'} \quad \text{(ResN)}$$

$$\frac{F \xrightarrow{\bar{x}\langle y\rangle} F'}{(\nu\, y : T)F \xrightarrow{(\nu\, y:T)\bar{x}\langle y\rangle} F'} \quad \text{(Scope)}$$

$$\frac{S \xrightarrow{l} S'}{(\nu\, R)S \xrightarrow{l} (\nu\, R)S'} \quad \text{(ResGS)}$$

$$\frac{P \xrightarrow{l} P'}{(\nu\, G)P\,\langle u\rangle \xrightarrow{l} (\nu\, G)P'\,\langle u\rangle} \quad \text{(ResGP)}$$

$$\frac{P \xrightarrow{l} P'}{!P \xrightarrow{l} P' \mid !P} \quad \text{(Repl)}$$

$$\frac{F_1 \xrightarrow{l_1} F_1' \qquad F_2 \xrightarrow{l_2} F_2' \qquad \mathrm{dual}(l_1, l_2)}{F_1 \mid F_2 \xrightarrow{\tau} (\nu\, \mathrm{bn}(l_1) \cup \mathrm{bn}(l_2))(F_1' \mid F_2')} \quad \text{(Com)}$$

$$\frac{P \xrightarrow{l} P'}{[x = x](P, P'') \xrightarrow{l} [\![x = x]\!]P'} \quad \text{(CondT)}$$

$$\frac{P \xrightarrow{l} P' \qquad x, y \in D_X \qquad x \neq y}{[x = y](P'', P) \xrightarrow{l} [\![x \neq y]\!]P'} \quad \text{(CondF)}$$

$$\frac{P \xrightarrow{l} P' \qquad op \in \{=, \neq\}}{[\![x\ op\ y]\!]P \xrightarrow{l} [\![x\ op\ y]\!]P'} \quad \text{(CondX)}$$

$$\frac{F_2 \xrightarrow{l} F \qquad F_1 \equiv_\alpha F_2}{F_1 \xrightarrow{l} F} \quad \text{(Alpha)}$$

Ο συμβολισμός $(\nu\, \mathrm{bn}(l_1) \cup \mathrm{bn}(l_2))$ στη μετάβαση (Com) βασίζεται στο ότι $\mathrm{dual}(l1, l2)$ και νοείται ως εξής:

$$(\nu\, \mathrm{bn}(l_1) \cup \mathrm{bn}(l_2)) \equiv \begin{cases} \epsilon & \text{αν } \{l_1, l_2\} = \{x(y : T), \bar{x}\,\langle y\rangle\} \\ (\nu\, y : T) & \text{αν } \{l_1, l_2\} = \{x(y : T), (\nu\, y : T)\bar{x}\,\langle y\rangle\} \end{cases},$$

όπου το ϵ συμβολίζει την κενή συμβολοσειρά.

3. Η σχέση $S \xrightarrow{l^*} S'$ δηλώνει ότι είτε $S = S'$ είτε το S' προκύπτει από το S με πεπερασμένο αριθμό μεταβάσεων.

Ο κανόνας (In) δηλώνει ότι μια διεργασία της μορφής $x(y : T).P$ μπορεί ανά πάσα στιγμή να λάβει ένα μήνυμα z στο κανάλι x και να συνεχίσει ως P, χρησιμοποιώντας το z όπου αναφερόταν το y. Ο κανόνας (Out) δηλώνει ότι μία διεργασία της μορφής $x\,\langle y\rangle\,.P$ μπορεί ανά πάσα στιγμή να στείλει το y μέσα από το κανάλι x. Ο κανόνας (ParL) δηλώνει ότι αν δύο διεργασίες τρέχουν παράλληλα και η πρώτη μπορεί από μόνη της να κάνει μια μετάβαση, τότε μπορεί –υπό την προϋπόθεση το ενδεχόμενο δεσμευμένο όνομα

της μετάβασης να μην εμφανίζεται ελεύθερο στη δεύτερη διεργασία– να την κάνει και τρέχοντας παράλληλα με τη δεύτερη. Ο κανόνας (ParR) δηλώνει το αντίστοιχο με τον (ParL) για τη δεύτερη διεργασία· αναγκαζόμαστε να αντιμετωπίσουμε τις δύο περιπτώσεις με διαφορετικούς κανόνες, επειδή για όσα αφορούν τις μεταβάσεις ταυτίζουμε μόνο τις α-ισοδύναμες διεργασίες και όχι τις όμοιες. Ο κανόνας (ResN) δηλώνει ότι η δέσμευση ενός ονόματος σε μια διεργασία ή ένα σύστημα διατηρεί τις μεταβάσεις, αν βέβαια το όνομα που δεσμεύεται δεν εμφανίζεται ελεύθερο στη μετάβαση. Ο κανόνας (Scope) καθορίζει πότε μια αποστολή μηνύματος μετατρέπεται σε δεσμευμένη αποστολή μηνύματος. Οι κανόνες (ResGP), (ResGS) και (CondX) δηλώνουν ότι τα αντίστοιχα συντακτικά στοιχεία δεν παίζουν κανένα ρόλο στις μεταβάσεις και διατηρούνται ως έχουν. Ο κανόνας (Repl) μοντελοποιεί τη διαίσθησή μας για το τι σημαίνει δυνάμει άπειρες επαναλήψεις: ένα αντίγραφο της P κάνει τη μετάβαση και παράλληλα εξακολουθεί να τρέχει το $!P$ για να παράξει άλλες επαναλήψεις. Ο κανόνας (Com) μοντελοποιεί την ανταλλαγή μηνυμάτων ανάμεσα σε επιμέρους διεργασίες ή συστήματα που τρέχουν παράλληλα. Οι κανόνες (CondF) και (CondT) μοντελοποιούν τον έλεγχο προϋποθέσεων: αν δύο ονόματα ταυτίζονται, τότε ακολουθούμε την πρώτη διεργασία, αλλιώς τη δεύτερη, σημειώνοντας το αποτέλεσμα του ελέγχου. Τέλος, ο (Alpha) δηλώνει ότι α-ισοδύναμες διεργασίες και α-ισοδύναμα συστήματα έχουν ακριβώς την ίδια συμπεριφορά ως προς τις μεταβάσεις· μπορούμε λοιπόν να ταυτίσουμε σημασιολογικά κάθε διεργασία και κάθε σύστημα με τις κλάσεις ισοδυναμίας τους σύμφωνα με τη σχέση $\equiv_\alpha$.

Να σημειωθεί εδώ πως ανά πάσα στιγμή ενδέχεται να είναι διαθέσιμες περισσότερες από μία δυνατές μεταβάσεις· το γεγονός αυτό συνεπάγεται πως η εξέλιξη των συστημάτων του π-λογισμού είναι μη ντετερμινιστική.

3.2.3 Παραδείγματα συστημάτων του π-λογισμού

Παράδειγμα 3.3. Συνεχίζουμε το παράδειγμα 3.2, δίνοντας μερικά συστήματα του π-λογισμού που μπορεί να χρησιμοποιούνται στην εταιρεία.

1. Κατά τη διάρκεια μιας αγοράς, το σύστημα S_{Alice} τρέχει στον φυλλομετρητή της Alice και προωθεί τη διεύθυνση του Bob στην εταιρεία. Ταυτόχρονα, τα συστήματα $S_{\text{PurchaseDpt}}$ και $S_{\text{ShippingDpt}}$ περιμένουν να δεχτούν τη διεύθυνση του Bob και να την επεξεργαστούν κατάλληλα: το μεν $S_{\text{PurchaseDpt}}$ τρέχει εκ μέρους του τμήματος αγορών, ελέγχει αν ο Bob είναι ενήλικας και σε θετική απάντηση προωθεί τη διεύθυνση στο τμήμα αποστολών, ενώ το $S_{\text{ShippingDpt}}$ τρέχει εκ μέρους του τμήματος αποστολών, δέχεται τη διεύθυνση και τη διαβάζει. Τα συστήματα $S_{\text{ShippingDpt}}$ και $S_{\text{PurchaseDpt}}$ συνεχίζουν παράλληλα να περιμένουν την

επόμενη παραγγελία. Συμβολίζουμε $T_1 = \text{Comp\&Clients}[\text{B.Address}]$.

$$S_{\text{Alice}} = (\nu\,\text{Alice})\overline{\text{sendaddr}}\,\langle\text{address}\rangle\,.\mathbf{0}\,\langle\text{purchase}\rangle$$

$$S_{\text{PurchaseDpt}} = (\nu\,\text{PurchaseDpt})!\text{getage}(\text{bobage} : \text{B.Age}).[\text{bobage} \neq 0 - 17]\ \text{sendaddr}(\text{addr} : T_1).\overline{\text{order}}\,\langle\text{addr}\rangle\,.\mathbf{0}\,\langle\text{purchase}\rangle$$

$$S_{\text{ShippingDpt}} = (\nu\,\text{ShippingDpt})!\text{order}(\text{addr} : T_1).\text{addr}(\text{a} : \text{B.Address}).\mathbf{0}\,\langle\text{purchase}\rangle$$

$$\begin{aligned} S_1 = (\nu\,\text{Comp\&Clients})(\nu\,\text{sendaddr} : \text{Comp\&Clients}[T_1])(\\ ((\nu\,\text{Clients})S_{\text{Alice}}) \mid ((\nu\,\text{Company})(\nu\,\text{OrderDpt})\\ (\nu\,\text{order} : \text{OrderDpt}[T_1])(\nu\,\text{getage} : \text{Company}[\text{B.Age}])(\\ S_{\text{PurchaseDpt}} \mid S_{\text{ShippingDpt}}\\ )\\ )\\ ) \end{aligned}$$

2. Το επόμενο σύστημα με το οποίο θα ασχοληθούμε τρέχει εκ μέρους του τμήματος προώθησης για διαφημιστικούς σκοπούς. Το σύστημα αυτό, έχοντας ως δεδομένο ότι ο Bob είναι ενήλικας, διαβάζει αν έχει δώσει τη συγκατάθεσή του και στέλνει τη διεύθυνσή του σε μία συνεργαζόμενη εταιρεία. Συμβολίζουμε $T_2 = \text{ThirdParty}[\text{ThirdParty}[\text{B.Address}]]$.

$$\begin{aligned} S_2 = (\nu\,\text{ThirdParty})(\nu\,\text{Comp\&Clients})(\nu\,\text{sendtotp} : T_2)(\nu\,\text{Company})\\ (\nu\,\text{MarketingDpt})[\![\text{bobage} \neq 0 - 17]\!]\text{readc}(\text{cons} : \text{B.Consent}).\\ (\nu\,\text{addr} : \text{ThirdParty}[\text{B.Address}])\overline{\text{sendtotp}}\,\langle\text{addr}\rangle\,.\mathbf{0}\,\langle\text{analysis}\rangle \end{aligned}$$

Παρατηρούμε ότι το σύστημα αυτό δεν φαίνεται να συμβαδίζει με την πολιτική ιδιωτικότητας της εταιρείας, διότι δεν διακόπτει την αποστολή της διεύθυνσης σε περίπτωση που την έχει αρνηθεί ο Bob· αυτό θα διαπιστωθεί και στον έλεγχο τύπων.

3. Η περίπτωση αυτή θα διασαφηνίσει τη λειτουργία των μεταβάσεων με ετικέτες.

   Κάποια στιγμή, το τμήμα διοίκησης αποφασίζει να συλλέξει και να επεξεργαστεί τα δεδομένα των χρηστών για να παράξει στατιστικές σχετικά με τη λειτουργία της εταιρείας. Γι' αυτό το λόγο, διαβάζει την ηλικία του Bob και, αφότου ελέγξει αν ο Bob είναι ενήλικας, συνεχίζει την επεξεργασία. Ταυτόχρονα, για το σκοπό της διατήρησης και διαμοιρασμού των δεδομένων εντός της εταιρείας, τρέχει ένα σύστημα σε ρόλο βάσης δεδομένων, το οποίο χρησιμοποιεί ένα δεδομένο κανάλι για να δίνει πρόσβαση στην ηλικία του Bob –η οποία τυχαίνει να είναι στο διάστημα 18-30– σε όποιον τη χρειάζεται. Συμβολίζουμε $T_3 = \text{B.Age}$ και $T_3' = \text{Company}[T_3]$.

$$\begin{aligned} P_{\text{DB}} &= !(\nu\,\text{age} : T_3')\overline{\text{getage}}\,\langle\text{age}\rangle\,.\overline{\text{age}}\,\langle 18 - 30\rangle\,.\mathbf{0}\\ S_{\text{DB}} &= (\nu\,\text{DB})P_{\text{DB}}\,\langle\text{disseminate}\rangle\\ S_{\text{AdminDpt}} &= (\nu\,\text{AdminDpt})\text{getage}(\text{readage} : T_3').\text{readage}(\text{y} : T_3).\\ &\qquad [\text{y} \neq 0 - 17]P\,\langle\text{analysis}\rangle\\ S_3 &= (\nu\,\text{Company})(\nu\,\text{getage} : \text{Company}[T_3'])(S_{\text{AdminDpt}} \mid S_{\text{DB}}) \end{aligned}$$

$$\overline{\text{getage}}\,\langle \text{age}\rangle\,.\overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \xrightarrow{\overline{\text{getage}}\langle \text{age}\rangle} \overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \quad \text{(Out)}$$

$$(\nu\,\text{age} : T_3')\overline{\text{getage}}\,\langle \text{age}\rangle\,.\overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \xrightarrow{(\nu\,\text{age}:T_3')\overline{\text{getage}}\langle \text{age}\rangle} \overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \quad \text{(Scope)}$$

$$P_{\text{DB}} \xrightarrow{(\nu\,\text{age}:T_3')\overline{\text{getage}}\langle \text{age}\rangle} \overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \mid P_{\text{DB}} \quad \text{(Repl)}$$

$$S_{\text{DB}} \xrightarrow{(\nu\,\text{age}:T_3')\overline{\text{getage}}\langle \text{age}\rangle} (\nu\,\text{DB})\overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \mid P_{\text{DB}}\,\langle \text{disseminate}\rangle \quad \text{(ResGP)}$$

$$\text{getage}(\text{readage} : T_3').\text{readage}(\text{y} : T_3).[\text{y} \neq 0-17]P \xrightarrow{\text{getage}(\text{age}:T_3')} \text{age}(\text{y} : T_3).[\text{y} \neq 0-17]P \quad \text{(In)}$$

$$S_{\text{AdminDpt}} \xrightarrow{\text{getage}(\text{age}:T_3')} (\nu\,\text{AdminDpt})\text{age}(\text{y} : T_3).[\text{y} \neq 0-17]P\,\langle \text{analysis}\rangle \quad \text{(ResGP)}$$

$$S_{\text{AdminDpt}} \mid S_{\text{DB}} \xrightarrow{\tau} S_3' \quad \text{(Com)}$$

$$S_3 \xrightarrow{\tau} (\nu\,\text{Company})(\nu\,\text{getage} : \text{Company}[T_3'])S_3' \quad \text{(ResN), (ResGS)}$$

όπου $S_3' = (\nu\,\text{AdminDpt})\text{age}(\text{y} : T_3).[\text{y} \neq 0-17]P\,\langle \text{analysis}\rangle \mid (\nu\,\text{DB})\overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \mid P_{\text{DB}}\,\langle \text{disseminate}\rangle$

$$\overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \xrightarrow{\overline{\text{age}}\langle 18-30\rangle} \mathbf{0} \quad \text{(Out)}$$

$$(\nu\,\text{DB})\overline{\text{age}}\,\langle 18-30\rangle\,.\mathbf{0} \mid P_{\text{DB}}\,\langle \text{disseminate}\rangle \xrightarrow{\overline{\text{age}}\langle 18-30\rangle} (\nu\,\text{DB})\mathbf{0} \mid P_{\text{DB}}\,\langle \text{disseminate}\rangle \equiv S_{\text{DB}} \quad \text{(ParL)}$$

$$\text{age}(\text{y} : T_3).[\text{y} \neq 0-17]P \xrightarrow{\text{age}(18-30:\text{B.Age})} [18-30 \neq 0-17]P \quad \text{(In)}$$

$$(\nu\,\text{AdminDpt})\text{age}(\text{y} : T_3).[\text{y} \neq 0-17]P\,\langle \text{analysis}\rangle \xrightarrow{\text{age}(18-30:\text{B.Age})} (\nu\,\text{AdminDpt})[18-30 \neq 0-17]P\,\langle \text{analysis}\rangle \quad \text{(ResGP)}$$

$$S_3' \xrightarrow{\tau} (\nu\,\text{AdminDpt})[18-30 \neq 0-17]P\,\langle \text{analysis}\rangle \mid S_{\text{DB}} \quad \text{(Com)}$$

$$(\nu\,\text{Company})(\nu\,\text{getage} : \text{Company}[T_3'])S_3' \xrightarrow{\tau} (\nu\,\text{Company})(\nu\,\text{getage} : \text{Company}[T_3'])((\nu\,\text{AdminDpt})[18-30 \neq 0-17]P\,\langle \text{analysis}\rangle \mid S_{\text{DB}}) \quad \text{(Com)}$$

Σχήμα 3.4: Δύο διαδοχικές μεταβάσεις του S_3.

Όπως βλέπουμε στο σχήμα 3.4, μετά από δύο σιωπηλές μεταβάσεις, το S_3 γίνεται

$$S_3'' = (\nu\,\text{Company})(\nu\,\text{getage} : \text{Company}[T_3'])(\\ (\nu\,\text{AdminDpt})[18-30 \neq 0-17]P\,\langle\text{analysis}\rangle \mid S_{\text{DB}}\\ ).$$

Παρατηρούμε επιπλέον ότι αν $P \xrightarrow{l} P'$, με παρόμοιο τρόπο (χρησιμοποιώντας και τον κανόνα (CondF)),

$$S_3'' \xrightarrow{l} (\nu\,\text{Company})(\nu\,\text{getage} : \text{Company}[T_3'])(\\ (\nu\,\text{AdminDpt})[\![18-30 \neq 0-17]\!]P'\,\langle\text{analysis}\rangle \mid S_{\text{DB}}\\ ).$$

3.3 Το σύστημα ελέγχου τύπων

Το σύστημα ελέγχου τύπων έχει ως στόχο να συνάγει τα δικαιώματα που απαιτεί μια διεργασία ή ένα σύστημα του π-λογισμού· ορίζεται στην πρώτη υποενότητα. Στη δεύτερη υποενότητα, αξιοποιούμε τη δυνατότητα αυτή του συστήματος ελέγχου τύπων για να ελέγξουμε αν ένα πρόγραμμα του π-λογισμού σέβεται μια πολιτική ιδιωτικότητας. Στην τρίτη υποενότητα, δίνουμε ένα παράδειγμα αυτής της διαδικασίας.

3.3.1 Ορισμοί για τον έλεγχο τύπων

Ορισμός 3.13. Το σύστημα ελέγχου τύπων αποτελείται από τρία βασικά στοιχεία:

1. Τα περιβάλλοντα Γ (Γ-Environments), τα οποία κάνουν τον έλεγχο τύπων του π-λογισμού, κρατώντας πληροφορίες σχετικές με την αντιστοίχιση των ονομάτων σε τύπους, καθώς και με τις ομάδες που έχουν εμφανιστεί σε ένα σύστημα.

   $$\Gamma \;::=\; \emptyset \quad | \quad x : T \quad | \quad G \quad | \quad \Gamma_1 \cdot \Gamma_2$$

   Κάθε όνομα και κάθε ομάδα πρέπει να εμφανίζονται σε ένα δεδομένο περιβάλλον Γ το πολύ μία φορά. Ορίζουμε τη συνάρτηση $\text{dom}(\Gamma)$, η οποία συλλέγει όλα τα ονόματα και τις ομάδες που εμφανίζονται στο Γ. Αν $\text{dom}(\Gamma) \subseteq \text{dom}(\Gamma')$ και τα Γ, Γ' αντιστοιχίζουν τα κοινά τους ονόματα στους ίδιους τύπους, γράφουμε $\Gamma \subseteq \Gamma'$.

2. Τις διεπαφές Θ (Θ-Interfaces), οι οποίες είναι τα στοιχεία εκείνα που καταγράφουν τις σχετικές με την ιδιωτικότητα πληροφορίες σε ένα σύστημα του π-λογισμού. Η σύνταξή τους είναι η ακόλουθη:

   $$\Theta \;::=\; \emptyset \quad | \quad t \gg \langle H^{\downarrow}, \tilde{p}\rangle \quad | \quad \Theta_1\,;\,\Theta_2 \;,$$

   όπου το $H^{\downarrow}$ καλείται ιεραρχία διεπαφής (Interface Hierarchy) και είναι της μορφής

   $$H^{\downarrow} \;::=\; G[u] \quad | \quad G[H^{\downarrow}] \;.$$

Ορίζουμε τη συνάρτηση $\mathrm{dom}(\Theta)$, η οποία συλλέγει όλους τους βασικούς τύπους που εμφανίζονται στο Θ και τη συνάρτηση $\mathrm{purpose}(H^{\downarrow})$, η οποία επιστρέφει το σκοπό που βρίσκεται στο $H^{\downarrow}$.

3. Τα περιβάλλοντα Δ (Δ-Environments), τα οποία καταγράφουν τις σχετικές με την ιδιωτικότητα πληροφορίες στις διεργασίες του πλογισμού.
$$\Delta \ ::= \ \emptyset \quad | \quad t : \tilde{p} \quad | \quad \Delta_1 \cdot \Delta_2$$

Κατά τη διάρκεια του ελέγχου τύπων, μία καταχώρηση της μορφής $x : T$ σε ένα περιβάλλον Γ σημαίνει ότι το όνομα x έχει τύπο T, ενώ μία καταχώρηση της μορφής G σημαίνει ότι η ομάδα G είναι «γνωστή» στο περιβάλλον Γ· οι καταχωρήσεις της μορφής G χρησιμεύουν για να αποτραπεί ο διαμοιρασμός πληροφοριών σε αυθαίρετες ομάδες. Μία διεπαφή Θ της μορφής $t \gg \langle H^{\downarrow}, \tilde{p} \rangle$ δηλώνει ότι ένα σύστημα τρέχει εκ μέρους κάποιου που ανήκει στις ομάδες της $H^{\downarrow}$ και ασκεί στο t τις ενέργειες που περιγράφονται στο $\tilde{p}$, για το σκοπό που εμφανίζεται στην ιεραρχία διεπαφής. Ένα περιβάλλον Δ της μορφής $t : \tilde{p}$ δηλώνει ότι μία διεργασία ασκεί στο t τις ενέργειες του $\tilde{p}$.

Σε αυτό το σημείο, θα ορίσουμε κάποιες βοηθητικές συναρτήσεις.

Ορισμός 3.14.

1. $$c \oplus p = \begin{cases} p' \text{ if } c \wedge c' & \text{αν } p = p' \text{ if } c' \\ p \text{ if } c & \text{διαφορετικά} \end{cases}$$
$$c \oplus \tilde{p} = \bigcup_{p \in \tilde{p}} c \oplus p$$

2. $$c \oplus \Delta = \begin{cases} (c \oplus \Delta_1) \cdot (c \oplus \Delta_2) & \text{αν } \Delta = \Delta_1 \cdot \Delta_2 \\ t : (c \oplus \tilde{p}) & \text{αν } \Delta = t : \tilde{p} \\ \emptyset & \text{διαφορετικά} \end{cases}$$

3. $$\Delta_{\mathrm{r}}(T) = \begin{cases} t : \mathtt{read} & \text{αν } T = t \\ t : \mathtt{access} & \text{αν } T = G[t] \\ \emptyset & \text{διαφορετικά} \end{cases}$$
$$\Delta_{\mathrm{w}}(T) = \begin{cases} t : \mathtt{write} & \text{αν } T = G[t] \\ t : \mathtt{disc}\, G & \text{αν } T = G[G'[t]] \\ \emptyset & \text{διαφορετικά} \end{cases}$$

4. $\Delta_1 \uplus \Delta_2 = \{t : \tilde{p_1} \cup \tilde{p_2} \mid t : \tilde{p_1} \in \Delta_1, t : \tilde{p_2} \in \Delta_2\}$, όπου για λόγους συντομίας της παρουσίασης υποθέτουμε ότι $t : \emptyset \in \Delta$ αν δεν υπάρχει άλλο $\tilde{p}$ ώστε $t : \tilde{p} \in \Delta$

5. $$G[u] \odot \Delta = \begin{cases} (G[u] \odot \Delta_1); (G[u] \odot \Delta_2) & \text{αν } \Delta = \Delta_1 \cdot \Delta_2 \\ t \gg \langle G[u], \tilde{p} \rangle & \text{αν } \Delta = t : \tilde{p} \\ \emptyset & \text{διαφορετικά} \end{cases}$$

6. $$G \odot \Theta = \begin{cases} (G \odot \Theta_1); (G \odot \Theta_2) & \text{αν } \Theta = \Theta_1; \Theta_2 \\ t \gg \langle G[H^{\downarrow}], \tilde{p} \rangle & \text{αν } \Theta = t \gg \langle H^{\downarrow}, \tilde{p} \rangle \\ \emptyset & \text{διαφορετικά} \end{cases}$$

Είμαστε τώρα έτοιμοι να ορίσουμε το σύστημα ελέγχου τύπων.

Ορισμός 3.15. Έχουμε τρεις σχέσεις που χρησιμοποιούνται για τον έλεγχο τύπων: την $\Gamma \vdash x \triangleright T$, η οποία δηλώνει ότι με βάση το περιβάλλον Γ το όνομα x έχει τύπο T, την $\Gamma \vdash P \triangleright \Delta$, η οποία δηλώνει ότι η διεργασία P είναι καλοδιατυπωμένη με βάση το περιβάλλον Γ και παράγει το περιβάλλον Δ και την $\Gamma \vdash S \triangleright \Theta$, η οποία δηλώνει ότι το σύστημα S είναι καλοδιατυπωμένο με βάση το περιβάλλον Γ και παράγει τη διεπαφή Θ. Ο έλεγχος τύπων γίνεται με βάση τους παρακάτω κανόνες:

$$\frac{\mathrm{fg}(T) \subseteq \mathrm{dom}(\Gamma)}{\Gamma \cdot x : T \vdash x \triangleright T} \quad \text{(Name)}$$

$$\Gamma \vdash \mathbf{0} \triangleright \emptyset \quad \text{(Nil)}$$

$$\frac{\Gamma \cdot y : T \vdash P \triangleright \Delta \qquad \Gamma \vdash x \triangleright G[T]}{\Gamma \vdash x(y : T).P \triangleright \Delta \uplus \Delta_{\mathrm{r}}(T)} \quad \text{(In)}$$

$$\frac{\Gamma \vdash P \triangleright \Delta \qquad \Gamma \vdash x \triangleright G[T] \qquad \Gamma \vdash y \triangleright T}{\Gamma \vdash \bar{x}\langle y \rangle .P \triangleright \Delta \uplus \Delta_{\mathrm{w}}(G[T])} \quad \text{(Out)}$$

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1 \qquad \Gamma \vdash P_2 \triangleright \Delta_2}{\Gamma \vdash P_1 \mid P_2 \triangleright \Delta_1 \uplus \Delta_2} \quad \text{(ParP)}$$

$$\frac{\Gamma \vdash S_1 \triangleright \Theta_1 \qquad \Gamma \vdash S_2 \triangleright \Theta_2}{\Gamma \vdash S_1 \mid S_2 \triangleright \Theta_1; \Theta_2} \quad \text{(ParS)}$$

$$\frac{\Gamma \cdot x : T \vdash P \triangleright \Delta}{\Gamma \vdash (\nu\, x : T)P \triangleright \Delta} \quad \text{(ResNP)}$$

$$\frac{\Gamma \cdot x : T \vdash S \triangleright \Theta}{\Gamma \vdash (\nu\, x : T)S \triangleright \Theta} \quad \text{(ResNS)}$$

$$\frac{\Gamma \cdot G \vdash P \triangleright \Delta}{\Gamma \vdash (\nu\, G)P\langle u \rangle \triangleright G[u] \odot \Delta} \quad \text{(ResGP)}$$

$$\frac{\Gamma \cdot R \vdash S \triangleright \Theta}{\Gamma \vdash (\nu\, R)S \triangleright R \odot \Theta} \quad \text{(ResGS)}$$

$$\frac{\Gamma \vdash P \triangleright \Delta}{\Gamma \vdash !P \triangleright \Delta} \quad \text{(Rep)}$$

$$\frac{\Gamma \vdash P \triangleright \Delta \qquad \Gamma \vdash x \triangleright X \qquad \Gamma \vdash y \triangleright X \qquad op \in \{=, \neq\}}{\Gamma \vdash [\![x\ op\ y]\!]P \triangleright (X\ op\ y) \oplus \Delta} \quad \text{(CondA)}$$

$$\frac{\Gamma \vdash [\![x = y]\!]P_1 \triangleright \Delta_1 \qquad \Gamma \vdash [\![x \neq y]\!]P_2 \triangleright \Delta_2}{\Gamma \vdash [x = y](P_1; P_2) \triangleright \Delta_1 \uplus \Delta_2} \quad \text{(CondB)}$$

Το αξίωμα (Nil) σημαίνει ότι η κενή διεργασία είναι καλώς διατυπωμένη και από αυτή συνάγεται το κενό περιβάλλον Δ, καθώς και ότι το κενό σύστημα είναι καλοδιατυπωμένο και από αυτό συνάγεται η κενή διεπαφή Θ. Ο κανόνας (Name) χρησιμοποιείται για να συνάγουμε τον τύπο των ονομάτων που παρουσιάζονται· για να είναι καλοδιατυπωμένο το πρόγραμμα, πρέπει οι ομάδες που εμφανίζονται στον τύπο του ονόματος να υπάρχουν ήδη στο περιβάλλον Γ. Ο κανόνας (In) δηλώνει ότι, αν οι τύποι του x και του y είναι τέτοιοι ώστε το x να είναι ένα κανάλι που μεταφέρει μηνύματα τύπου y και αν επιπλέον η διεργασία P με βάση το περιβάλλον Γ $\Gamma \cdot y : T$ παράγει το περιβάλλον Δ Δ, τότε η $x(y : T).P$ με βάση το Γ παράγει το Δ, επεκτεταμένο κατά t : `read` αν $T = t$, κατά t : `access` αν $T = G[t]$ και μη επεκτεταμένο για οποιοδήποτε άλλο T. Ο κανόνας (Out) δηλώνει ότι, αν οι τύποι του x

και του y είναι τέτοιοι ώστε το x να είναι ένα κανάλι που δέχεται μηνύματα τύπου y και αν επιπλέον η διεργασία P με βάση το περιβάλλον Γ $\Gamma \cdot y : T$ παράγει το περιβάλλον Δ Δ, τότε η $\bar{x}\langle y \rangle).P$ με βάση το Γ παράγει το Δ, επεκτεταμένο κατά t : `write` αν ο τύπος του x είναι της μορφής $G[t]$, κατά t : `disc` G αν ο τύπος του x είναι της μορφής $G[G'[t]]$ και μη επεκτεταμένο για οποιοδήποτε άλλο T. Ο κανόνας (ParP) δηλώνει ότι αν οι διεργασίες P_1 και P_2 είναι καλοδιατυπωμένες και παράγουν τα Δ_1 και Δ_2 αντίστοιχα, τότε η $P_1 \mid P_2$ είναι καλοδιατυπωμένη και παράγει το $\Delta_1 \uplus \Delta_2$. Ο κανόνας (ParS) δηλώνει ότι αν τα συστήματα S_1 και S_2 είναι καλοδιατυπωμένα και παράγουν τις Θ_1 και Θ_2 αντίστοιχα, τότε το $S_1 \mid S_2$ είναι καλοδιατυπωμένο και παράγει το $\Theta_1; \Theta_2$. Ο κανόνας (ResNP) (αντ. (ResNS)) δηλώνει ότι η δέσμευση μίας μεταβλητής σε μία διεργασία (αντ. ένα σύστημα) είναι καλοδιατυπωμένη με βάση ένα περιβάλλον Γ και παράγει ένα περιβάλλον Δ (αντ. μια διεπαφή Θ) αν παραλείποντας τη δέσμευση και προσθέτοντάς την ως καταχώρηση στο περιβάλλον Γ, η διεργασία (αντ. το σύστημα) είναι καλοδιατυπωμένη και παράγει το ίδιο περιβάλλον Δ (αντ. διεπαφή Θ). Ο κανόνας (ResGP) δηλώνει ότι αν η διεργασία P είναι καλοδιατυπωμένη με βάση το περιβάλλον Γ $\Gamma \cdot G$ και παράγει το περιβάλλον Δ Δ, τότε το σύστημα $(\nu\, G)P\,[u]$ είναι καλοδιατυπωμένο με βάση το περιβάλλον Γ και παράγει τη διεπαφή Θ $G[u] \odot \Delta$. Ο κανόνας (ResGS) δηλώνει ότι αν το σύστημα S είναι καλοδιατυπωμένο με βάση το περιβάλλον Γ $\Gamma \cdot R$ και παράγει τη διεπαφή Θ Θ, τότε το σύστημα $(\nu\, R)S$ είναι καλοδιατυπωμένο με βάση το περιβάλλον Γ και παράγει τη διεπαφή Θ $G \odot \Theta$. Ο κανόνας (Rep) δηλώνει ότι η επανάληψη μιας καλοδιατυπωμένης διεργασίας είναι καλοδιατυπωμένη και παράγει το ίδιο περιβάλλον Δ. Ο κανόνας (CondA) δηλώνει ότι αν η διεργασία P είναι καλοδιατυπωμένη με βάση το περιβάλλον Γ Γ και παράγει το περιβάλλον Δ Δ και τα ονόματα x και y έχουν τον ίδιο τύπο, ο οποίος μάλιστα είναι μια μεταβλητή πλαισίου X, τότε η διεργασία $[\![x\ op\ y]\!]P$, με $op \in \{=, \neq\}$, είναι καλοδιατυπωμένη με βάση το ίδιο περιβάλλον Γ και παράγει το περιβάλλον Δ που προκύπτει προσθέτοντας στο Δ την ατομική προϋπόθεση $(X\ op\ y)$. Ο κανόνας (CondB) δηλώνει ότι αν η διεργασία $[\![x\ =\ y]\!]P_1$ είναι καλοδιατυπωμένη με βάση το περιβάλλον Γ Γ και παράγει το περιβάλλον Δ Δ_1 και η διεργασία $[\![x\ =\ y]\!]P_2$ είναι καλοδιατυπωμένη με βάση το περιβάλλον Γ Γ και παράγει το περιβάλλον Δ Δ_2, τότε η διεργασία $[x = y](P_1; P_2)$ είναι καλοδιατυπωμένη με βάση το ίδιο περιβάλλον Γ και παράγει το περιβάλλον $\Delta \uplus \Delta_2$.

Οι κανόνες (CondA) και (CondB) μπορούν να συνδυαστούν και να δώσουν τον κανόνα

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1 \qquad \Gamma \vdash P_2 \triangleright \Delta_2 \qquad \Gamma \vdash x \triangleright X \qquad \Gamma \vdash y \triangleright X}{\Gamma \vdash [x = y](P_1; P_2) \triangleright ((X\ =\ y) \oplus \Delta_1) \uplus ((X\ \neq\ y) \oplus \Delta_2)} \quad \text{(Cond)}$$

3.3.2 Συμβατότητα συστήματος με πολιτική

Έχοντας πλέον τη δυνατότητα να βρούμε τι δικαιώματα απαιτεί ένα σύστημα του π-λογισμού, είναι φυσικό να θέλουμε να κάνουμε το ερώτημα αν τα δικαιώματα που απαιτεί είναι συμβατά με την επιθυμητή πολιτική (και επομένως, έμμεσα, αν το σύστημα είναι συμβατό με την πολιτική).

Ορισμός 3.16. Λέμε ότι μια διεπαφή Θ ικανοποιεί μια πολιτική ιδιωτικό-

τητας και συμβολίζουμε $\mathcal{P} \vDash \Theta$, αν

$$\forall t \gg \left\langle H^{\downarrow}, \tilde{p} \right\rangle \in \Theta \, \exists t \gg H, \pi \in \mathcal{P} : \tilde{p} \lesssim \mathrm{perms}_{\pi}(H, \mathrm{groups}(H^{\downarrow}), \mathrm{purpose}(H^{\downarrow})) \ ,$$

όπου

$$\mathrm{perms}_{\pi}(H, \tilde{G}, u) = \begin{cases} \pi(u, G) \cup \left(\bigcup\limits_{\substack{H_i \in \tilde{H} \\ \mathrm{root}(H_i) \subseteq \tilde{G}}} \mathrm{perms}_{\pi}(u \to H_i, \mathrm{groups}(H_i) \cap \tilde{G}, u) \right) & \\ & \text{αν } H = G : \tilde{u}[\tilde{H}], G \in \tilde{G}, u \in \tilde{u} \\ \bigcup\limits_{\substack{H_i \in \tilde{H} \\ \mathrm{root}(H_i) \subseteq \tilde{G}}} \mathrm{perms}_{\pi}(H_i, \mathrm{groups}(H_i) \cap \tilde{G}, u) & \\ & \text{αν } H = G : \tilde{u}[\tilde{H}], G \in \tilde{G}, u \notin \tilde{u} \\ \emptyset & \text{αν } \tilde{G} = \emptyset \text{ ή } H = \epsilon \\ \bot & \text{αλλιώς} \end{cases}$$

Η συνάρτηση $\mathrm{perms}_{\pi}(H, \tilde{G}, u)$ διασχίζει όλα τα μονοπάτια που ξεκινούν από τη ρίζα της ιεραρχίας H και περιέχουν ομάδες από το $\tilde{G}$· καθώς διασχίζει την ιεραρχία, συλλέγει τα δικαιώματα που παραχωρούνται για το σκοπό u στις ομάδες του $\tilde{G}$ που θα συναντήσει. Όπως έχουμε αναφέρει, αν σε κάποια ομάδα G παραχωρείται ο σκοπός u, τότε η παραχώρηση ισχύει άρρητα και για όλες τις ομάδες που βρίσκονται «κάτω» από την G στην ιεραρχία· αυτό δηλώνεται ρητά στον ορισμό της $\mathrm{perms}_{\pi}(\cdot, \cdot, \cdot)$. Σε περίπτωση που η ρίζα της ιεραρχίας δεν βρίσκεται στο $\tilde{G}$, η συνάρτηση επιστρέφει $\bot$.

Ισχυρισμός. Η συνάρτηση $\mathrm{perms}_{\pi}(\cdot, \cdot, \cdot)$ τερματίζει.

Απόδειξη. Παρατηρούμε ότι κάθε φορά που καλείται εκ νέου η συνάρτηση, το δεύτερο όρισμα έχει αυστηρώς λιγότερα στοιχεία, διότι το έχουμε τμήσει με το $\mathrm{groups}(H_i)$, το οποίο –λόγω της εξ ορισμού απουσίας κύκλων στις ιεραρχίες– δεν περιέχει τη ρίζα G της H· η G όμως βρίσκεται στο $\tilde{G}$. Επομένως, επειδή ασχολούμαστε με πεπερασμένα σύνολα, στη χειρότερη περίπτωση κάποια στιγμή θα καταλήξουμε στο κενό σύνολο, για το οποίο η συνάρτηση τερματίζει.

Προφανώς, υπάρχουν και άλλοι τρόποι να τερματίσει η συνάρτηση πριν καταλήξει σε κενό σύνολο ομάδων, αλλά δεν χρειάζεται να τους εξετάσουμε για αυτή την απόδειξη. □

Η ακόλουθη παρατήρηση προκύπτει εύκολα εξετάζοντας τον ορισμό:

Παρατήρηση 3.17. Αν η συνάρτηση $\mathrm{perms}_{\pi}(H, \tilde{G}, u)$ δεν επιστρέφει $\bot$, τότε

$$\mathrm{perms}_{\pi}(H, \tilde{G}, u) \subseteq \bigcup_{G \in \tilde{G}} \pi(u, G) \ .$$

Έχουμε λοιπόν τη δυνατότητα να ελέγξουμε το κατά πόσο ένα σύστημα του π-λογισμού σέβεται μια πολιτική ιδιωτικότητας. Θα θέλαμε ο έλεγχος αυτός να γίνεται αυτόματα, αλλά, σύμφωνα με τα παραπάνω, χρειάζεται να καθορίσουμε ένα περιβάλλον Γ με βάση το οποίο θα γίνει ο έλεγχος τύπων· είναι προφανές ότι υπάρχουν άπειρες επιλογές περιβαλλόντων Γ, κάποια

από τα οποία οδηγούν σε θετικό και κάποια σε αρνητικό αποτέλεσμα. Η ίδια όμως η δομή του συστήματος μπορεί να δίνει πληροφορίες για τους τύπους μερικών ονομάτων· η παρατήρηση αυτή μας οδηγεί στη διατύπωση του παρακάτω ορισμού, ο οποίος –όπως θα αποδείξουμε αργότερα– κατασκευάζει το «ελάχιστο» περιβάλλον Γ που μπορούμε να χρησιμοποιήσουμε:

Ορισμός 3.18. Έστω σύστημα S. Κατασκευάζουμε το περιβάλλον Γ $\Gamma_\emptyset(S)$, ως εξής: για κάθε $x \in \text{fn}(S)$,

- Αν $\exists X \in \mathcal{X} : x \in D_X$, προσθέτουμε στο $\Gamma_\emptyset(S)$ μια καταχώρηση $x : X$.
- Αν το x εμφανίζεται μέσα στο S σε ένα στοιχείο της μορφής $[\![x \; op \; y]\!]P$ ή της μορφής $[x = y](P_1; P_2)$ και $y \in D_X$, προσθέτουμε στο $\Gamma_\emptyset(S)$ μια καταχώρηση $x : X$.
- Αν το x εμφανίζεται μέσα στο S σε ένα στοιχείο της μορφής $\overline{z}\langle x \rangle .P$ και το z έχει δεσμευτεί στο S με τύπο $G[T]$, προσθέτουμε στο $\Gamma_\emptyset(S)$ μια καταχώρηση $x : T$.
- Αν το x εμφανίζεται μέσα στο S σε ένα στοιχείο της μορφής $\overline{z}\langle x \rangle .P$ και το $\Gamma_\emptyset(S)$ περιέχει καταχώρηση $z : G[T]$, προσθέτουμε στο $\Gamma_\emptyset(S)$ μια καταχώρηση $x : T$.

Παρατηρούμε πως, λόγω του τελευταίου σημείου, η κατασκευή του $\Gamma_\emptyset(S)$ μπορεί να χρειαστεί περισσότερες από μία επαναλήψεις για να ολοκληρωθεί· επειδή όμως το S έχει πεπερασμένο μήκος, κάποια στιγμή θα ολοκληρωθεί. Από το λήμμα 3.28, τους κανόνες ελέγχου τύπων και την κατασκευή του $\Gamma_\emptyset(S)$ προκύπτει η ακόλουθη συνέπεια:

Συνέπεια. $\Gamma \vdash S \triangleright \Theta \;\Rightarrow\; \Gamma_\emptyset(S) \subseteq \Gamma$

Ορισμός 3.19. Λέμε ότι ένα σύστημα S είναι συμβατό με (ή σέβεται ή ικανοποιεί) μια πολιτική ιδιωτικότητας $\mathcal{P}$ (συμβολισμός $\mathcal{P} \vDash S$) αν για κάθε περιβάλλον Γ Γ με $\text{fn}(S) \subseteq \text{dom}(\Gamma)$ και $\Gamma_\emptyset(S) \subseteq \Gamma$, έχουμε $\Gamma \vdash S \triangleright \Theta$ και $\mathcal{P} \vDash \Theta$. Σε περίπτωση που το προηγούμενο ισχύει για το $\Gamma_\emptyset(S)$, γράφουμε $\mathcal{P} \vdash S$ και λέμε πως το S περνάει τον έλεγχο τύπων.

3.3.3 Παραδείγματα χρήσης του συστήματος ελέγχου τύπων

Παράδειγμα 3.4. Συνεχίζουμε το παράδειγμα 3.3, δίνοντας το αποτέλεσμα του ελέγχου τύπων στα δύο πρώτα συστήματα του π-λογισμού που παρουσιάστηκαν εκεί. Σε όλες τις περιπτώσεις, συμβολίζουμε $c_0 = (\text{B.Age} \neq 0 - 17)$.

1. Για το S_1, έχουμε $\Gamma_1 = \Gamma_\emptyset(S_1) = 0 - 17 : \text{B.Age} \cdot \text{address} : T_1$. Συμβολίζουμε $\Gamma_{\text{Alice}} = \Gamma_1 \cdot \text{sendaddr} : \text{Comp\&Clients}[T_1] \cdot \text{Comp\&Clients} \cdot \text{Clients}$, $\Gamma_{\text{PD}} = \Gamma_1 \cdot \text{sendaddr} : \text{Comp\&Clients}[T_1] \cdot \text{Comp\&Clients} \cdot \text{Company} \cdot \text{OrderDpt} \cdot \text{order} : \text{OrderDpt}[T_1] \cdot \text{PurchaseDpt}$, $\Gamma'_{\text{PD}} = \Gamma_{\text{PD}} \cdot \text{PurchaseDpt}$.

   Στο σχήμα 3.5 βλέπουμε πώς γίνεται ο έλεγχος τύπων στο S_1, καταλήγοντας στη διεπαφή Θ Θ_1. Διαπιστώσουμε έπειτα ότι το S_1 σέβεται την πολιτική ιδιωτικότητας: για το

   $$\text{B.Address} \gg \langle \text{Comp\&Clients}[\text{Clients}[\text{Alice}[\text{purchase}]]], \{\texttt{disc}\ \text{Comp\&Clients}\} \rangle$$

$$\frac{\Gamma_{\text{Alice}} \cdot \text{Alice} \vdash \mathbf{0}}{\Gamma_{\text{Alice}} \cdot \text{Alice} \vdash \overline{\text{sendaddr}}\,\langle \text{address} \rangle\,.\mathbf{0}} \quad \frac{}{\triangleright \emptyset} \text{(Nil)}$$

$$\frac{\Gamma_{\text{Alice}} \cdot \text{Alice} \vdash \overline{\text{sendaddr}}\,\langle \text{address} \rangle\,.\mathbf{0}}{\Gamma_{\text{Alice}} \vdash S_{\text{Alice}}} \quad \frac{\triangleright \emptyset}{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{Comp\&Clients}\}} \text{(Out)}$$

$$\frac{\Gamma_{\text{Alice}} \vdash S_{\text{Alice}}}{\Gamma_1 \cdot \text{Comp\&Clients} \cdot \text{sendaddr} : \text{Comp\&Clients}[T_1] \vdash (\nu\,\text{Clients})S_{\text{Alice}}} \quad \frac{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{Comp\&Clients}\}}{\triangleright \text{B.Address} \gg \langle \text{Alice}, \{\texttt{disc}\ \text{Comp\&Clients}\}\rangle} \text{(ResGP)}$$

$$\frac{\Gamma_1 \cdot \text{Comp\&Clients} \cdot \text{sendaddr} : \text{Comp\&Clients}[T_1] \vdash (\nu\,\text{Clients})S_{\text{Alice}}}{\Gamma'_{\text{PD}} \cdot \text{addr} : T_1 \vdash \mathbf{0}} \quad \frac{\triangleright \text{B.Address} \gg \langle \text{Alice}, \{\texttt{disc}\ \text{Comp\&Clients}\}\rangle}{\triangleright \text{B.Address} \gg \langle \text{Clients[Alice]}, \{\texttt{disc}\ \text{Comp\&Clients}\}\rangle} \text{(ResGS)}$$

$$\frac{\Gamma'_{\text{PD}} \cdot \text{addr} : T_1 \vdash \mathbf{0}}{\Gamma'_{\text{PD}} \cdot \text{addr} : T_1 \vdash \overline{\text{order}}\,\langle \text{addr} \rangle\,.\mathbf{0}} \quad \frac{\triangleright \emptyset}{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{OrderDpt}\}} \text{(Out)}$$

$$\frac{\Gamma'_{\text{PD}} \cdot \text{addr} : T_1 \vdash \overline{\text{order}}\,\langle \text{addr} \rangle\,.\mathbf{0}}{\Gamma'_{\text{PD}} \vdash \text{sendaddr}(\text{addr} : T_1).\overline{\text{order}}\,\langle \text{addr} \rangle\,.\mathbf{0}} \quad \frac{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{OrderDpt}\}}{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{OrderDpt}, \texttt{access}\}} \text{(In)}$$

$$\frac{\Gamma'_{\text{PD}} \vdash \text{sendaddr}(\text{addr} : T_1).\overline{\text{order}}\,\langle \text{addr} \rangle\,.\mathbf{0}}{\Gamma'_{\text{PD}} \vdash [\text{bobage} \neq 0 - 17]\text{sendaddr}(\text{addr} : T_1).\overline{\text{order}}\,\langle \text{addr} \rangle\,.\mathbf{0}} \quad \frac{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{OrderDpt}, \texttt{access}\}}{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{OrderDpt if } c_0, \texttt{access if } c_0\}} \text{(Cond)}$$

$$\frac{\Gamma'_{\text{PD}} \vdash [\text{bobage} \neq 0 - 17]\text{sendaddr}(\text{addr} : T_1).\overline{\text{order}}\,\langle \text{addr} \rangle\,.\mathbf{0}}{\Gamma'_{\text{PD}} \vdash\, !\,[\text{bobage} \neq 0 - 17]\text{sendaddr}(\text{addr} : T_1).\overline{\text{order}}\,\langle \text{addr} \rangle\,.\mathbf{0}} \quad \frac{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{OrderDpt if } c_0, \texttt{access if } c_0\}}{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{OrderDpt if } c_0, \texttt{access if } c_0\}} \text{(Rep)}$$

$$\frac{\Gamma'_{\text{PD}} \vdash\, !\,[\text{bobage} \neq 0 - 17]\text{sendaddr}(\text{addr} : T_1).\overline{\text{order}}\,\langle \text{addr} \rangle\,.\mathbf{0}}{\Gamma_{\text{PD}} \vdash S_{\text{PurchaseDpt}}} \quad \frac{\triangleright \text{B.Address} : \{\texttt{disc}\ \text{OrderDpt if } c_0, \texttt{access if } c_0\}}{\triangleright \text{B.Address} \gg \langle\, \text{PuchaseDpt[purchase]}, \{\texttt{disc}\ \text{OrderDpt if } c_0, \texttt{access if } c_0\}\,\rangle} \text{(ResGS)}$$

Συνεχίζοντας με αυτόν τον τρόπο, καταλήγουμε ότι $\Gamma_1 \vdash S_1 \triangleright \Theta_1$, με

$$\begin{aligned}\Theta_1 = \ & \text{B.Address} \gg \langle \text{Comp\&Clients[Clients[Alice[purchase]]]}, \{\texttt{disc}\ \text{Comp\&Clients}\}\rangle \\ & ;\text{B.Address} \gg \langle \text{Comp\&Clients[Company[OrderDpt[PurchaseDpt[purchase]]]]}, \{\texttt{access if } c_0, \texttt{disc}\ \text{OrderDpt if } c_0\}\rangle \\ & ;\text{B.Address} \gg \langle \text{Comp\&Clients[Company[OrderDpt[ShippingDpt[purchase]]]]}, \{\texttt{access}, \texttt{read}\}\rangle\end{aligned}$$

Σχήμα 3.5: Ένα ενδεικτικό κομμάτι και το αποτέλεσμα του ελέγχου τύπων στο σύστημα S_1.

έχουμε ότι

$$\mathrm{perms}_{\mathcal{P}_{\mathrm{Bob}},\Gamma_1}(H, \{\text{Comp\&Clients}, \text{Clients}, \text{Alice}\}, \text{purchase}) = \{\texttt{read}, \texttt{access}, \texttt{disc}\ \text{Comp\&Clients}\}$$

και προφανώς $\{\texttt{disc}\ \text{Comp\&Clients}\} \lesssim \{\texttt{read}, \texttt{access}, \texttt{disc}\ \text{Comp\&Clients}\}$· όμοια αντιμετωπίζονται και οι άλλες δύο συνιστώσες του Θ_1, άρα καταλήγουμε ότι $\mathcal{P}_{\mathrm{Bob}} \vDash \Theta_1$ και επομένως $\mathcal{P}_{\mathrm{Bob}} \vdash S_1$. Ειδικότερα, επειδή $\Gamma_1 = \Gamma_\emptyset(S_1)$, έχουμε $\mathcal{P}_{\mathrm{Bob}} \vdash^{\mathrm{t}} S_1$.

2. Όμοια με πριν, διεξάγουμε τον έλεγχο τύπων και καταλήγουμε ότι $\Gamma_2 \vdash S_2 \triangleright \Theta_2$, με

$$\begin{aligned}\Gamma_2 &= \text{age} : \text{B.Age} \cdot 0 - 17 : \text{B.Age} \cdot \text{readc} : \text{Comp\&Clients}[\text{B.Consent}],\\ \Theta_2 &= \ \text{B.Address} \gg \langle \text{Comp\&Clients}[\text{ThirdParty}[\text{Company}[\text{MarketingDpt}[\\ &\qquad \text{marketing}]]]], \{\texttt{disc}\ \text{ThirdParty if } c_0\}\rangle\\ &\quad ;\ \text{B.Consent} \gg \langle \text{Comp\&Clients}[\text{ThirdParty}[\text{Company}[\text{MarketingDpt}[\\ &\qquad \text{marketing}]]]], \{\texttt{read}\ \text{if } c_0\}\rangle\end{aligned}$$

Έχουμε ότι

$$\mathrm{perms}_{\pi_{\text{B.Address}}}(H, \{\text{Comp\&Clients}, \text{ThirdParty}, \text{Company}, \text{MarketingDpt}\}, \text{marketing}) = \{\texttt{access}\ \text{if } c_0, \texttt{disc}\ \text{ThirdParty if } c_0 \wedge c_1\}$$

και $\{\texttt{disc}\ \text{ThirdParty if } c_0\} \not\lesssim \{\texttt{access}\ \text{if } c_0, \texttt{disc}\ \text{ThirdParty if } c_0 \wedge c_1\}$, άρα καταλήγουμε ότι $\mathcal{P}_{\mathrm{Bob}} \nvDash \Theta_2$. Επομένως, $\mathcal{P}_{\mathrm{Bob}} \nvDash S_2$.

3.4 Αποδείξεις ορθότητας

Το γεγονός ότι ένα πρόγραμμα περνά τον έλεγχο τύπων δεν μας λέει από μόνο του κάτι για την ορθότητά του· με την έννοια «ορθότητα», εννοούμε εδώ ότι αποκλείεται είτε ένα πρόγραμμα είτε η εξέλιξή του μετά από οσεσδήποτε μεταβάσεις –δηλαδή, διαισθητικά, μετά από οσεσδήποτε ανταλλαγές μηνυμάτων ανάμεσα στα επιμέρους στοιχεία του– να προσπαθήσει να ασκήσει ένα δικαίωμα που δεν προβλέπεται από την πολιτική ιδιωτικότητας. Θα δώσουμε μια άμεση απόδειξη της ορθότητας του συστήματος ελέγχου τύπων μας, στα πρότυπα του [19]. Δεν θα αποδείξουμε τη γενική περίπτωση· θα δείξουμε όμως ότι αν το πρόγραμμα έχει γραφτεί ώστε να περιέχει ενδείξεις για τους τύπους όλων των κρίσιμων δεδομένων του, τότε ο έλεγχος τύπων αποδεικνύει την ορθότητα –ή μη– του προγράμματος.

Αρχικά, πρέπει να ορίσουμε αυστηρά το πότε ένα πρόγραμμα επιχειρεί να ασκήσει ένα δικαίωμα που δεν του έχει παραχωρηθεί. Θα χρησιμοποιήσουμε την έννοια του σφάλματος, το οποίο είναι ένα σύστημα που είτε είναι κακοδιατυπωμένο είτε στην αμέσως επόμενη μετάβαση θα (υπάρχει περίπτωση να) παραβιάσει την πολιτική ιδιωτικότητας. Αυτή η σχετικά αδύναμη έννοια αρκεί για να διατυπώσουμε και να αποδείξουμε το θεώρημα της ορθότητας του συστήματός μας.

Ορισμός 3.20.

1. Έστω πολιτική ιδιωτικότητας $\mathcal{P}$, περιβάλλον Γ Γ και σύστημα S.

   Θεωρούμε το συντακτικό δέντρο του S και σε αυτό εντοπίζουμε όλες τις διεργασίες που είναι της μορφής $P = x(y : T).P'$ ή $P = \overline{x}\langle y\rangle.P'$ και κανένας πρόγονός τους στο δέντρο δεν είναι της ίδιας μορφής ή της μορφής $[a = v](P_1; P_2)$.

   Το S θα καλείται σφάλμα σε σχέση με τα $\mathcal{P}$ και Γ (συμβολισμός $\mathrm{error}_{\mathcal{P},\Gamma}(S)$), αν δεν υπάρχει Θ ώστε $\Gamma \vdash S \triangleright \Theta$ ή αν υπάρχει $t \in \mathcal{D}$ ώστε $\mathcal{P} = t \gg H, \pi; \mathcal{P}'$ και για κάποια από τις διεργασίες που έχουμε εντοπίσει ισχύει κάτι από τα παρακάτω:

   (α′) $P = x(y : t).P'$ και $\left\{\texttt{read if } \left(\vec{X}\ \vec{op}\ \vec{v}\right)\right\} \not\subseteq \bigcup_{G\in\tilde{G}} \pi(u, G)$

   (β′) $P = x\langle y\rangle.P'$, $\Gamma' \vdash y \triangleright t$ και $\left\{\texttt{write if } \left(\vec{X}\ \vec{op}\ \vec{v}\right)\right\} \not\subseteq \bigcup_{G\in\tilde{G}} \pi(u, G)$

   (γ′) $P = x(y : G'[t]).P'$ και $\left\{\texttt{access if } \left(\vec{X}\ \vec{op}\ \vec{v}\right)\right\} \not\subseteq \bigcup_{G\in\tilde{G}} \pi(u, G)$

   (δ′) $P = x\langle y\rangle.P'$, $\Gamma' \vdash x \triangleright G'[G''[t]]$
   και $\left\{\texttt{disc}\, G' \text{ if } \left(\vec{X}\ \vec{op}\ \vec{v}\right)\right\} \not\subseteq \bigcup_{G\in\tilde{G}} \pi(u, G)$

   όπου $\tilde{G}$ είναι οι δεσμευμένες ομάδες στην εμβέλεια των οποίων βρίσκεται η P, Γ' είναι η επέκταση του Γ με τις δεσμευμένες ομάδες στην εμβέλεια των οποίων βρίσκεται η P και με τα δεσμευμένα ονόματα στην εμβέλεια των οποίων βρίσκεται η P, ενώ τα $\left\{p \text{ if } \left(\vec{X}\ \vec{op}\ \vec{v}\right)\right\}$ προκύπτουν ως εξής: για κάθε $[\![z\ op\ v]\!]$ στην εμβέλεια του οποίου βρίσκεται η P, προσθέτουμε στο p την προϋπόθεση $(X\ op\ v)$, με $\Gamma' \vdash z \triangleright X$.

2. Έστω πολιτική ιδιωτικότητας $\mathcal{P}$ και σύστημα S. Το S θα καλείται σφάλμα σε σχέση με την $\mathcal{P}$ (συμβολισμός $\mathrm{error}_{\mathcal{P}}(S)$), αν υπάρχει περιβάλλον Γ Γ με $\mathrm{fn}(S) \subseteq \Gamma$ ώστε το S είναι σφάλμα σε σχέση με τα $\mathcal{P}$ και Γ.

Πρόταση 3.21. $\mathrm{error}_{\mathcal{P},\Gamma}(S) \ \wedge\ \Gamma \vdash S \triangleright \Theta \ \Rightarrow\ \mathcal{P} \not\models \Theta$

Απόδειξη. Από τις περιπτώσεις (1α′), (1β′), (1γ′) και (1δ′) του $\mathrm{error}_{\mathcal{P},\Gamma}(S)$. Θα εξετάσουμε αναλυτικά την περίπτωση (1α′)· η απόδειξη των υπολοίπων είναι όμοια.

Έχουμε $t \in \mathcal{D}$ και διάνυσμα μεταβλητών πλαισίου $\vec{Y}$ ώστε $\mathcal{P} = t \gg H, \pi; \mathcal{P}'$, $\Gamma \vdash \vec{y} \triangleright \vec{Y}$ και επιπλέον $P = x(y : t).P''$ και $\left\{\texttt{read if } \left(\vec{Y}\ \vec{op}\ \vec{z}\right)\right\} \not\subseteq \bigcup_{i=1}^{n} \pi(u, G_i)$.

Κατά τη διάρκεια του ελέγχου τύπων, θα χρειαστεί να εφαρμόσουμε τον κανόνα (In) στη διεργασία P, παίρνοντας $\Gamma' \vdash P \triangleright \Delta' \uplus t : \texttt{read}$ για κάποιο Δ' που εξαρτάται από τη δομή του P'. Έπειτα, ακολουθώντας το συντακτικό δέντρο του S μέχρι τη ρίζα του, θα οφείλουμε να χρησιμοποιήσουμε τους κανόνες (CondA), (Rep), (ResNP), (ParP), (ResGP), (ResGS), (ParS) και (ResNS) για να προχωρήσει ο έλεγχος τύπων· σε όλους αυτούς το δικαίωμα $t : \texttt{read}$

θα παραμένει στο περιβάλλον Δ, εκτός από τον (CondA), σε κάθε εφαρμογή του οποίου θα προσθέτουμε μία προϋπόθεση του $[\![\vec{y}\ \vec{op}\ \vec{z}]\!]$ στο t : `read`. Θα καταλήξουμε έτσι στο $\Gamma \vdash S \triangleright \Theta$, με $\Theta = t \gg \langle G_1[G_2[\cdots G_n[u]]], \tilde{p}\rangle ; \Theta'$, όπου `read if` $\left(\vec{Y}\ \vec{op}\ \vec{z}\right) \in \tilde{p}$ και επομένως $\tilde{p} \not\lesssim \bigcup_{i=1}^{n} \pi(u, G_i)$.

Από τις παρατηρήσεις 3.17 και 3.4 ισχύει ότι $\mathrm{perms}_\pi(H, \{G_1, \ldots, G_n\}, u) \lesssim \bigcup_{i=1}^{n} \pi(u, G_i)$, άρα, λόγω της μεταβατικής ιδιότητας της $\lesssim$, παίρνουμε $\tilde{p} \not\lesssim \mathrm{perms}_\pi(H, \{G_1, \ldots, G_n\}, u)$ και επομένως, εξ ορισμού, $\mathcal{P} \not\Vdash \Theta$. □

Όπως αναφέρθηκε ήδη, ο ορισμός του σφάλματος ασχολείται μόνο με την αμέσως επόμενη ενέργεια που μπορεί να ασκήσει ένα σύστημα. Προφανώς, θα μπορούσε μία από τις επόμενες ενέργειές του να παραβιάσει την πολιτική ιδιωτικότητας· θα αποδείξουμε ότι –αν το σύστημα περνάει τον έλεγχο τύπων– κάτι τέτοιο δεν είναι εφικτό.

Αρχικά, επεκτείνουμε τους ορισμούς που έχουμε δώσει ώστε να λαμβάνουν υπόψιν την «ιστορία» ενός συστήματος. Η τροποποιήσεις βασίζονται στο παρακάτω λήμμα:

Λήμμα 3.22.

1. $P \xrightarrow{l} P' \ \Rightarrow\ \mathrm{fn}(P') \subseteq \mathrm{fn}(P) \cup \mathrm{tn}(l)$
2. $S \xrightarrow{l} S' \ \Rightarrow\ \mathrm{fn}(S') \subseteq \mathrm{fn}(S) \cup \mathrm{tn}(l)$

Απόδειξη. Παρατηρώντας τους κανόνες της σημασιολογίας των μεταβάσεων. □

Ορισμός 3.23. Έστω πολιτική $\mathcal{P}$, σύστημα S και σύνολο ετικετών $\tilde{l}$, το οποίο θα ερμηνεύουμε ως τις μεταβάσεις που έχει ήδη κάνει το S. Ορίζουμε τότε τα παρακάτω:

- $\Gamma_{\tilde{l}}(S) \overset{\text{ορ}}{=} \Gamma_\emptyset(S) \cup \left\{ y : T \mid x(y:T) \in \tilde{l} \ \vee\ (\nu\, y : T)\overline{x}\langle y\rangle \in \tilde{l} \right\}$
- $\mathcal{P} \vDash_{\tilde{l}} S$ αν για κάθε περιβάλλον Γ Γ με $\mathrm{fn}(S) \cup \left(\bigcup_{l\in\tilde{l}} \mathrm{tn}(l)\right) \subseteq \mathrm{dom}(\Gamma)$ και $\Gamma_{\tilde{l}}(S) \subseteq \Gamma$ έχουμε $\Gamma \vdash S \triangleright \Theta$ και $\mathcal{P} \vDash \Theta$. Αν το προηγούμενο ισχύει για το $\Gamma_{\tilde{l}}(S)$, γράφουμε $\mathcal{P} \vdash_{\tilde{l}} S$.
- $\mathrm{error}_{\mathcal{P},\tilde{l}}(S)$ αν υπάρχει περιβάλλον Γ Γ με $\mathrm{fn}(S) \cup \left(\bigcup_{l\in\tilde{l}} \mathrm{tn}(l)\right) \subseteq \mathrm{dom}(\Gamma)$ για το οποίο $\mathrm{error}_{\mathcal{P},\Gamma}(S)$.

Θα χρειαστεί τώρα να ορίσουμε μία (προ)διάταξη $\lesssim$ (διαβάζεται «απαιτεί λιγότερα δικαιώματα από») στα στοιχεία του ελέγχου τύπων και να δείξουμε μερικές χρήσιμες ιδιότητές της.

Ορισμός 3.24.

1. $\Delta_1 \lesssim \Delta_2 \overset{\text{ορ}}{\Leftrightarrow} \forall t : \tilde{p} \in \Delta_1 \exists t : \tilde{p}' \in \Delta_2 : \tilde{p} \lesssim \tilde{p}'$
2. $\Theta_1 \lesssim \Theta_2 \overset{\text{ορ}}{\Leftrightarrow} \mathrm{dom}(\Theta_1) = \mathrm{dom}(\Theta_2)$
   $\wedge\ \forall t \gg \langle H^{\downarrow}, \tilde{p}\rangle \in \Theta_1 \exists t \gg \langle H^{\downarrow}, \tilde{p}'\rangle \in \Theta_2 : \tilde{p} \lesssim \tilde{p}'$

Πρόταση 3.25. *Οι $\lesssim$ του ορισμού 3.24 είναι προδιατάξεις.*

Απόδειξη. Τελείως ανάλογα με την απόδειξη της περίπτωσης 3 της πρότασης 3.6. □

Παρατήρηση 3.26.

1. Η πράξη $\uplus$ ανάμεσα σε περιβάλλοντα Δ σέβεται την προδιάταξη $\lesssim$, δηλαδή $\Delta_1 \lesssim \Delta_2 \;\Rightarrow\; \Delta_1 \uplus \Delta_0 \lesssim \Delta_2 \uplus \Delta_0$.
2. Η πράξη $\oplus$ ανάμεσα σε ατομικές προϋποθέσεις και περιβάλλοντα Δ σέβεται την προδιάταξη $\lesssim$, δηλαδή $\Delta_1 \lesssim \Delta_2 \;\Rightarrow\; (X\ op\ y) \oplus \Delta_1 \lesssim (X\ op\ y) \oplus \Delta_2$.
3. $\Theta_1 \lesssim \Theta_2 \;\Rightarrow\; \Theta_1;\Theta_0 \lesssim \Theta_2;\Theta_0$
4. $\Delta_1 \lesssim \Delta_2 \;\Rightarrow\; G[u] \oplus \Delta_1 \lesssim G[u] \oplus \Delta_2$
5. $\Theta_1 \lesssim \Theta_2 \;\Rightarrow\; G \oplus \Theta_1 \lesssim G \oplus \Theta_2$

Απόδειξη.

1. Έστω $\Delta_1 \lesssim \Delta_2$ και ένα στοιχείο $t : \tilde{p} \in \Delta_1 \uplus \Delta_0$. Τότε, υπάρχουν $t : \tilde{p}_1 \in \Delta_1$ και $t : \tilde{p}_0 \in \Delta_0$ ώστε $\tilde{p} = \tilde{p}_1 \cup \tilde{p}_0$. Επειδή $\Delta_1 \lesssim \Delta_2$, υπάρχει $t : \tilde{p}_2 \in \Delta_2$ με $\tilde{p}_1 \lesssim \tilde{p}_2$, άρα (εξ ορισμού) $\tilde{p}_1 \cup \tilde{p}_0 \lesssim \tilde{p}_2 \cup \tilde{p}_0$, και επειδή υπάρχει $t : \tilde{p}_2 \in \Delta_2$, έχουμε $t : \tilde{p}_2 \cup \tilde{p}_0 \in \Delta_2 \uplus \Delta_0$. Επομένως, $\Delta_1 \uplus \Delta_0 \lesssim \Delta_2 \uplus \Delta_0$.
2. Έστω $t : \tilde{p} \in (X\ op\ y) \oplus \Delta_1$. Τότε, κάθε δικαίωμα $p \in \tilde{p}$ είναι της μορφής $(X\ op\ y) \oplus p_1$. Επειδή $\Delta_1 \lesssim \Delta_2$, θα υπάρχουν $t : \tilde{p}' \in \Delta_2$ και $p_2 \in \tilde{p}'$ με $p_1 \leq p_2$. Από τον ορισμό της $\leq$ ανάμεσα σε δικαιώματα, παίρνουμε $(X\ op\ y) \oplus p_1 \leq (X\ op\ y) \oplus p_2$. Επομένως, $\{(X\ op\ y) \oplus p_1\} \lesssim (X\ op\ y) \oplus \tilde{p}'$, από όπου έχουμε το ζητούμενο.
3. Προφανώς, αν $\mathrm{dom}(\Theta_1) = \mathrm{dom}(\Theta_2)$, τότε $\mathrm{dom}(\Theta_1;\Theta_0) = \mathrm{dom}(\Theta_2;\Theta_0)$. Έστω $\Theta_1 \lesssim \Theta_2$ και ένα στοιχείο $t \gg \langle H^{\downarrow}, \tilde{p}\rangle \in \Theta_1;\Theta_0$. Τότε είτε $t \gg \langle H^{\downarrow}, \tilde{p}\rangle \in \Theta_1$ είτε $t \gg \langle H^{\downarrow}, \tilde{p}\rangle \in \Theta_0$. Στη δεύτερη περίπτωση, έχουμε τετριμμένα το ζητούμενο. Στην πρώτη περίπτωση, έχουμε το ζητούμενο επειδή $\Theta_1 \lesssim \Theta_2$.
4. Έστω $t \gg \langle H^{\downarrow}, \tilde{p}_1\rangle \in G[u] \oplus \Delta_1$. Τότε $H^{\downarrow} = G[u]$. Επειδή $\Delta_1 \lesssim \Delta_2$, έχουμε ότι υπάρχει $\tilde{p}_2 \in \Delta_2$ με $\tilde{p}_1 \lesssim \tilde{p}_2$ και μάλιστα $t \gg \langle H^{\downarrow}, \tilde{p}_2\rangle \in G[u] \oplus \Delta_2$, από όπου προκύπτει το ζητούμενο.
5. Όμοια με το προηγούμενο.

□

Πρόταση 3.27. $\mathcal{P} \vDash \Theta_1 \;\wedge\; \Theta_2 \lesssim \Theta_1 \;\Rightarrow\; \mathcal{P} \vDash \Theta_2$

Απόδειξη. Εξ ορισμού, το $\Theta_2 \lesssim \Theta_1$ συνεπάγεται ότι $\forall t \gg \langle H^{\downarrow}, \tilde{p}_2\rangle \in \Theta_2 \exists t \gg \langle H^{\downarrow}, \tilde{p}_1\rangle \in \Theta_1 : \tilde{p}_2 \lesssim \tilde{p}_1$, όπου $H^{\downarrow} = G_1[G_2[\ldots G_n[u]\ldots]]$. Επειδή όμως $\mathcal{P} \vDash \Theta_1$, έχουμε ότι $\exists t \gg H, \pi \in \mathcal{P} : \tilde{p}_1 \lesssim \mathrm{perms}_{\pi}(H, \{G_1, G_2, \ldots, G_n\}, u)$. Οπότε, από τη μεταβατική ιδιότητα της $\lesssim$ (πρόταση 3.6), $\forall t \gg \langle H^{\downarrow}, \tilde{p}_2\rangle \in \Theta_2 \exists t \gg H, \pi \in \mathcal{P} : \tilde{p}_2 \lesssim \mathrm{perms}_{\pi}(H, \{G_1, G_2, \ldots, G_n\}, u)$, δηλαδή $\mathcal{P} \vDash \Theta_2$. □

Θα αποδείξουμε τώρα μερικά λήμματα τα οποία είναι χρήσιμα και διαλευκαίνουν τις ιδιότητες των περιβαλλόντων Γ.

Λήμμα 3.28.

1. $\Gamma \vdash P \triangleright \Delta \;\Rightarrow\; \mathrm{fn}(P) \cup \mathrm{fg}(P) \subseteq \mathrm{dom}(\Gamma)$
2. $\Gamma \vdash S \triangleright \Theta \;\Rightarrow\; \mathrm{fn}(S) \cup \mathrm{fg}(S) \subseteq \mathrm{dom}(\Gamma)$

Απόδειξη. Με επαγωγή στη δομή του συστήματος/διεργασίας. Έστω $\Gamma_P \vdash P \triangleright \Delta$ και $\Gamma_S \vdash S \triangleright \Theta$.

- Για την κενή διεργασία και το κενό σύστημα, το ζητούμενο ισχύει τετριμμένα.
- Έστω $P = \overline{x}\langle y \rangle.P'$. Ο κανόνας (Out) απαιτεί να ισχύει $\Gamma_P \vdash x \triangleright G[T]$, από το οποίο –λόγω του κανόνα (Name)– παίρνουμε ότι $x \in \Gamma_P$, και $\Gamma_P \vdash y \triangleright T$, από το οποίο –λόγω του κανόνα (Name)– παίρνουμε ότι $y \in \Gamma_P$. Από την επαγωγική υπόθεση, $\mathrm{fn}(P') \cup \mathrm{fg}(P') \subseteq \mathrm{dom}(\Gamma_P)$, άρα, επειδή $\mathrm{fn}(P) = \mathrm{fn}(P') \cup \{x, y\}$ και $\mathrm{fg}(P) = \mathrm{fg}(P')$, έχουμε το ζητούμενο.
- Αν $P = [\![x \; op \; y]\!]P'$, η απόδειξη γίνεται τελείως όμοια με την προηγούμενη περίπτωση.
- Έστω $P = P_1 \mid P_2$. Από την επαγωγική υπόθεση, $\mathrm{fn}(P_1) \cup \mathrm{fg}(P_1) \subseteq \mathrm{dom}(\Gamma_P)$ και $\mathrm{fn}(P_2) \cup \mathrm{fg}(P_2) \subseteq \mathrm{dom}(\Gamma_P)$. Επειδή $\mathrm{fn}(P) = \mathrm{fn}(P_1) \cup \mathrm{fn}(P_2)$ και $\mathrm{fg}(P) = \mathrm{fg}(P_1) \cup \mathrm{fg}(P_2)$, έχουμε το ζητούμενο.
- Αν $S = S_1 \mid S_2$ ή $P = [x = y](P_1; P_2)$ ή $P = !P'$, η απόδειξη γίνεται τελείως όμοια με την προηγούμενη περίπτωση.
- Έστω $P = (\nu\, x : T)P'$. Από την επαγωγική υπόθεση, $\mathrm{fn}(P') \cup \mathrm{fg}(P') \subseteq \mathrm{dom}(\Gamma'_P)$, με $\Gamma'_P = \Gamma_P \cdot x : T$. Επειδή $\mathrm{fn}(P) = \mathrm{fn}(P') \setminus \{x\}$, $\mathrm{fg}(P) = \mathrm{fg}(P')$ και $\mathrm{dom}(\Gamma_P) = \mathrm{dom}(\Gamma'_P) \setminus \{x\}$, έχουμε το ζητούμενο.
- Αν $S = (\nu\, x : T)S'$ ή $S = (\nu\, G)S'$ ή $S = (\nu\, G)P\langle u \rangle$, η απόδειξη γίνεται τελείως όμοια με την προηγούμενη περίπτωση.
- Αν $P = x(y : T).P'$, η απόδειξη είναι συνδυασμός των περιπτώσεων $P = (\nu\, x : T)P'$ και $P = \overline{x}\langle y \rangle.P'$.

□

Λήμμα 3.29. *Αν το x δεν είναι τιμή μεταβλητής πλαισίου,*

1. $\Gamma \cdot x : T \vdash P \triangleright \Delta \;\wedge x, y \notin \mathrm{bn}(P) \;\Rightarrow\; \Gamma \cdot y : T \vdash P\{y/x\} \triangleright \Delta$
2. $\Gamma \cdot x : T \vdash S \triangleright \Theta \;\wedge x, y \notin \mathrm{bn}(S) \;\Rightarrow\; \Gamma \cdot y : T \vdash S\{y/x\} \triangleright \Theta$

Απόδειξη. Με επαγωγή στη δομή του συστήματος ή της διεργασίας.

- Για την κενή διεργασία και το κενό σύστημα, το ζητούμενο ισχύει τετριμμένα από τον κανόνα (Nil) του ελέγχου τύπων.
- Αν $P = !P'$, τότε ο μόνος κανόνας του ελέγχου τύπων από τον οποίον μπορεί να έχει προκύψει το $\Gamma \cdot x : T \vdash P \triangleright \Delta$ είναι ο (Rep), ο οποίος προϋποθέτει ότι $\Gamma \cdot x : T \vdash P' \triangleright \Delta$. Από την επαγωγική υπόθεση, $\Gamma \cdot y : T \vdash P'\{y/x\} \triangleright \Delta$. Εφαρμόζοντας τον κανόνα (Rep), έχουμε $\Gamma \cdot y : T \vdash !(P'\{y/x\}) \triangleright \Delta$ και άρα $\Gamma \cdot y : T \vdash !P'\{y/x\} \triangleright \Delta$.

- Όμοια με την προηγούμενη περίπτωση αποδεικνύονται οι $P = P_1 \mid P_2$, $S = S_1 \mid S_2$, $P = \overline{x}\langle y\rangle.P'$, $P = [\![z \ op \ v]\!]P'$, $P = [x = y](P_1; P_2)$, $S = (\nu\, R)S'$ και $S = (\nu\, G)P\langle u\rangle$.

- Αν $P = (\nu\, z : T')P'$, τότε ο μόνος κανόνας του ελέγχου τύπων από τον οποίον μπορεί να έχει προκύψει το $\Gamma \cdot x : T \vdash P \triangleright \Delta$ είναι ο (ResNP), ο οποίος προϋποθέτει ότι $\Gamma \cdot x : T \cdot z : T' \vdash P' \triangleright \Delta$. Από την επαγωγική υπόθεση, $\Gamma \cdot y : T \cdot z : T' \vdash P'\{y/x\} \triangleright \Delta$ και από τον (ResNP) προκύπτει το ζητούμενο.
  Επειδή $x, y \notin \mathrm{bn}(P)$, έχουμε ότι $x \neq z$ και $y \neq z$, άρα όλα τα περιβάλλοντα Γ είναι καλώς ορισμένα.

- Τελείως όμοια με την προηγούμενη περίπτωση αποδεικνύονται οι $S = (\nu\, z : T')S'$ και $P = b(z : T').P'$.

□

Πρόταση 3.30. *Ο έλεγχος τύπων έχει το ίδιο αποτέλεσμα σε κάθε δύο α-ισοδύναμες διεργασίες ή συστήματα, δηλαδή*

$$\Gamma \vdash P_1 \triangleright \Delta \ \wedge \ P_1 \equiv_\alpha P_2 \ \Rightarrow \ \Gamma \vdash P_2 \triangleright \Delta$$

και

$$\Gamma \vdash S_1 \triangleright \Theta \ \wedge \ S_1 \equiv_\alpha S_2 \ \Rightarrow \ \Gamma \vdash S_2 \triangleright \Theta$$

Απόδειξη. Με επαγωγή στη δομή του συστήματος/διεργασίας. Έστω $\Gamma \vdash P_1 \triangleright \Delta$ και $\Gamma_S \vdash S_1 \triangleright \Theta$.

- Το κενό σύστημα και η κενή διεργασία είναι α-ισοδύναμα μόνο με τον εαυτό τους, άρα το ζητούμενο ισχύει τετριμμένα.

- Αν $P_1 = x(y : T).P$, τότε $P_2 = x(z : T).P'$, με τα y, z να μην είναι τιμές μεταβλητών πλαισίου και με $P\{b/y\} \equiv_\alpha P'\{b/z\}$ για κατάλληλο b. Από τον κανόνα (In) του ελέγχου τύπων, $\Gamma \cdot y : T \vdash P \triangleright \Delta'$, με $\Delta = \Delta' \uplus \Delta_{\mathrm{r}}(T)$. Από το λήμμα 3.29, παίρνουμε $\Gamma \cdot b : T \vdash P\{b/y\} \triangleright \Delta'$, άρα από την επαγωγική υπόθεση $\Gamma \cdot b : T \vdash \triangleright \Delta'$. Μπορούμε να θεωρήσουμε ότι $z \notin \mathrm{bn}(P')$ (διότι διαφορετικά θα εκμεταλλευτούμε την α-ισοδυναμία και την επαγωγική υπόθεση), οπότε το λήμμα 3.29 δίνει $\Gamma \cdot z : T \vdash P' \triangleright \Delta'$ και από τον κανόνα (In) του ελέγχου τύπων $\Gamma \vdash P_2 \triangleright \Delta$.

- Όμοια αποδεικνύονται οι περιπτώσεις $P_1 = (\nu\, x : T)P$ και $S_1 = (\nu\, x : T)S$.

- Αν $P_1 = \overline{x}\langle y\rangle.P$, τότε $P_2 = \overline{x}\langle y\rangle.P'$, με $P \equiv_\alpha P'$. Από τον κανόνα (Out) του ελέγχου τύπων, $\Gamma \vdash P \triangleright \Delta'$, με $\Delta = \Delta' \uplus \Delta_{\mathrm{w}}(G[T])$. Από την επαγωγική υπόθεση, $\Gamma \vdash P' \triangleright \Delta'$ και εφαρμόζοντας πάλι τον κανόνα (Out) παίρνουμε το ζητούμενο.

- Οι υπόλοιπες περιπτώσεις είναι όμοιες με την παραπάνω.

□

Συνέπεια 3.31. *Αν το x δεν είναι τιμή μεταβλητής πλαισίου,*

1. $\Gamma \cdot x : T \vdash P \triangleright \Delta \ \Rightarrow \ \Gamma \cdot y : T \vdash P\{y/x\} \triangleright \Delta$

2. $\Gamma \cdot x : T \vdash S \triangleright \Theta \;\Rightarrow\; \Gamma \cdot y : T \vdash S\{y/x\} \triangleright \Theta$

Λήμμα 3.32.

1. $\Gamma \cdot z : T \vdash x \triangleright T \;\wedge z \neq x \;\Rightarrow\; \Gamma \vdash x \triangleright T$
2. $\Gamma \cdot z : T \vdash P \triangleright \Delta \;\wedge z \notin \mathrm{fn}(P) \Rightarrow\; \Gamma \vdash P \triangleright \Delta$
3. $\Gamma \cdot z : T \vdash S \triangleright \Theta \;\wedge z \notin \mathrm{fn}(S) \Rightarrow\; \Gamma \vdash S \triangleright \Theta$

Απόδειξη.

1. Άμεσα, από τον κανόνα (Name) του ελέγχου τύπων.
2. Με επαγωγή στη δομή της διεργασίας. Έστω $\Gamma \cdot z : T \vdash P \triangleright \Delta$ με $z \notin \mathrm{fn}(P)$.
   - Για την κενή διεργασία, το ζητούμενο ισχύει τετριμμένα λόγω του κανόνα (Nil).
   - Έστω $P = \overline{x}\langle y \rangle .P'$. Ο κανόνας (Out) του ελέγχου τύπων απαιτεί να ισχύει $\Gamma \cdot z : T \vdash x \triangleright G[T]$, $\Gamma \cdot z : T \vdash y \triangleright T$ και $\Gamma \cdot z : T \vdash P' \triangleright \Delta'$. Επειδή $z \notin \mathrm{fn}(P)$, έχουμε $z \neq x$ και $z \neq y$, από τα οποία παίρνουμε $\Gamma \vdash x \triangleright G[T]$ και $\Gamma \vdash y \triangleright T$. Επίσης, από την επαγωγική υπόθεση, $\Gamma \vdash P' \triangleright \Delta'$. Τέλος, εφαρμόζουμε τον κανόνα (Out) και παίρνουμε $\Gamma \vdash P \triangleright \Delta$.
   - Αν $P = [\![x \; op \; y]\!]P'$, η απόδειξη γίνεται τελείως όμοια με την προηγούμενη περίπτωση.
   - Έστω $P = P_1 \mid P_2$. Ο κανόνας (ParP) του ελέγχου τύπων απαιτεί να ισχύει $\Gamma \cdot z : T \vdash P_1 \triangleright \Delta_1$ και $\Gamma \cdot z : T \vdash P_2 \triangleright \Delta_2$. Από την επαγωγική υπόθεση, $\Gamma \vdash P_1 \triangleright \Delta_1$ και $\Gamma \vdash P_2 \triangleright \Delta_2$, οπότε, εφαρμόζοντας τον κανόνα (ParP), παίρνουμε το ζητούμενο.
   - Αν $P = !P'$, η απόδειξη γίνεται τελείως όμοια με την προηγούμενη περίπτωση.
   - Αν $P = [x = y](P_1; P_2)$, η απόδειξη είναι συνδυασμός των πιο πάνω περιπτώσεων.
   - Έστω $P = (\nu\, x : T')P'$. Χωρίς βλάβη της γενικότητας, έχουμε $x \neq z$ (αν $x = z$, τότε επιλέγουμε μια α-ισοδύναμη διεργασία με άλλο όνομα στη θέση του x). Ο κανόνας (ResNP) του ελέγχου τύπων απαιτεί $\Gamma \cdot z : T \cdot x : T' \vdash P' \triangleright \Delta$, από το οποίο μέσω της επαγωγικής υπόθεσης παίρνουμε $\Gamma \cdot x : T' \vdash P' \triangleright \Delta$. Εφαρμόζοντας τον κανόνα (ResNP), έχουμε το ζητούμενο.
   - Αν $P = x(y : T').P'$, η απόδειξη είναι συνδυασμός των περιπτώσεων $P = (\nu\, x : T)P'$ και $P = \overline{x}\langle y \rangle .P'$.
3. Τελείως όμοια με τις αποδείξεις για διεργασίες.

□

Λήμμα 3.33.

1. $\Gamma \vdash x \triangleright T \;\wedge\; y \notin \mathrm{dom}(\Gamma) \;\Rightarrow \Gamma \cdot y : T \vdash x \triangleright T$

2. $\Gamma \vdash P \triangleright \Delta \ \wedge \ y \notin \mathrm{dom}(\Gamma) \ \Rightarrow \Gamma \cdot y : T \vdash P \triangleright \Delta$

3. $\Gamma \vdash S \triangleright \Theta \ \wedge \ y \notin \mathrm{dom}(\Gamma) \ \Rightarrow \Gamma \cdot y : T \vdash S \triangleright \Theta$

Απόδειξη.

1. Άμεση, από τον κανόνα (Name) του ελέγχου τύπων.

2. Με επαγωγή στη δομή της P. Έστω $\Gamma \vdash P \triangleright \Delta$ και $y \notin \mathrm{dom}(\Gamma)$. Χωρίς βλάβη της γενικότητας, υποθέτουμε ότι $y \notin \mathrm{bn}(P)$ (αν έχουμε $y \in \mathrm{bn}(P)$, χρησιμοποιούμε την πρόταση 3.30 και επιλέγουμε μια διεργασία α-ισοδύναμη με την P, μετονομάζοντας το y σε κάποιο όνομα $z \notin \mathrm{dom}(\Gamma) \cup \mathrm{bn}(P) \cup \mathrm{fn}(P)$).

   - Αν $P = \mathbf{0}$, τότε ισχύει τετριμμένα, λόγω του κανόνα (Nil).
   - Αν $P = (\nu\, x : T')P'$, τότε ο μόνος κανόνας του ελέγχου τύπων από τον οποίο μπορεί να έχει προκύψει το $\Gamma \vdash P \triangleright \Delta$ είναι ο (ResNP), ο οποίος προϋποθέτει ότι $\Gamma \cdot x : T' \vdash P' \triangleright \Delta$. Επειδή $y \notin \mathrm{bn}(P)$, παίρνουμε $y \neq x$ και επομένως $y \notin \mathrm{dom}(\Gamma \cdot x : T')$. Μπορούμε λοιπόν να εφαρμόσουμε την επαγωγική υπόθεση, από την οποία παίρνουμε ότι $\Gamma \cdot x : T' \cdot y : T \vdash P' \triangleright \Delta$ και, από τον κανόνα (ResNP), $\Gamma \cdot y : T \vdash P \triangleright \Delta$.
   - Αν $P = x(y : T).P'$,τότε η απόδειξη είναι όμοια με της προηγούμενης περίπτωσης.
   - Αν $P = !P'$, τότε ο μόνος κανόνας του ελέγχου τύπων από τον οποίο μπορεί να έχει προκύψει το $\Gamma \vdash P \triangleright \Delta$ είναι ο (Rep), που προϋποθέτει ότι $\Gamma \vdash P' \triangleright \Delta$. Από την επαγωγική υπόθεση, $\Gamma \cdot y : T \vdash P' \triangleright \Delta$, άρα, από τον κανόνα (Rep), $\Gamma \cdot y : T \vdash P \triangleright \Delta$.
   - Οι υπόλοιπες περιπτώσεις αποδεικνύονται τελείως όμοια με την προηγούμενη.

3. Τελείως όμοια με την περίπτωση των διεργασιών.

□

Θεώρημα 3.34.

1. $\Gamma \vdash P_1 \triangleright \Delta \ \wedge \ P_1 \xrightarrow{l} P_2 \ \Rightarrow \ \Gamma' \vdash P_2 \triangleright \Delta' \ \wedge \ \Delta' \lesssim \Delta$

2. $\Gamma \vdash S_1 \triangleright \Theta \ \wedge \ S_1 \xrightarrow{l} S_2 \ \Rightarrow \ \Gamma' \vdash S_2 \triangleright \Theta' \ \wedge \ \Theta' \lesssim \Theta$

όπου $\Gamma' = \begin{cases} \Gamma \cdot y : T & \text{αν } y \notin \mathrm{dom}(\Gamma) \wedge (l = x(y : T) \ \vee \ l = (\nu\, y : T)\overline{x}\,\langle y \rangle) \\ \Gamma & \text{αλλιώς} \end{cases}$

Απόδειξη. Εξετάζουμε πώς μπορεί να γίνει η μετάβαση σε κάθε περίπτωση:

1. Έστω $\Gamma \vdash P_1 \triangleright \Delta$ και $P_1 \xrightarrow{l} P_2$.

   (In) Έχουμε $P_1 = x(y : T).P$, $P_2 = P\{z/y\}$ και $l = x(z : T)$. Ξέρουμε ότι $\Gamma \vdash x(y : T).P \triangleright \Delta$, το οποίο μπορεί να έχει προκύψει μόνο από τον κανόνα (In) του ελέγχου τύπων, ο οποίος προϋποθέτει ότι $\Gamma \vdash x \triangleright G[T]$ και $\Gamma \cdot y : T \vdash P \triangleright \Delta'$, όπου $\Delta = \Delta' \uplus \Delta_{\mathrm{r}}(T)$, από το οποίο εξ ορισμού προκύπτει ότι $\Delta' \lesssim \Delta$.

- Αν $z \in \mathrm{dom}(\Gamma)$, τότε, επειδή τα y, z έχουν τον ίδιο τύπο, παίρνουμε $\Gamma \cdot y : T \vdash P\{z/y\} \triangleright \Delta'$ και από το λήμμα 3.32 $\Gamma \vdash P_2 \triangleright \Delta'$.
- Αν $z \notin \mathrm{dom}(\Gamma)$, από τη συνέπεια 3.31, $\Gamma \cdot z : T \vdash P_2 \triangleright \Delta'$.

(Out) Έχουμε $P_1 = \bar{x}\langle y \rangle.P_2$ και $l = \bar{x}\langle y \rangle$. Ξέρουμε ότι $\Gamma \vdash \bar{x}\langle y \rangle.P_2 \triangleright \Delta$, το οποίο μπορεί να έχει προκύψει μόνο από τον κανόνα (Out) του ελέγχου τύπων, ο οποίος προϋποθέτει ότι $\Gamma \vdash x \triangleright G[T]$ και $\Gamma \vdash P_2 \triangleright \Delta'$, όπου $\Delta = \Delta' \uplus \Delta_{\mathrm{w}}(G[T])$, από το οποίο εξ ορισμού προκύπτει ότι $\Delta' \lesssim \Delta$.

(Scope) Έχουμε $P_1 = (\nu\, y : T)P$, $l = (\nu\, y : T)\bar{x}\langle y \rangle$ και $P \xrightarrow{x\langle y \rangle} P_2$. Το $\Gamma \vdash (\nu\, y : T)P \triangleright \Delta$ μπορεί να προκύψει μόνο από τον κανόνα (ResNP), ο οποίος προϋποθέτει ότι $\Gamma \cdot y : T \vdash P \triangleright \Delta$. Από την επαγωγική υπόθεση, $\Gamma \cdot y : T \vdash P_2 \triangleright \Delta'$, με $\Delta' \lesssim \Delta$.

(Com) Έχουμε $P_1 = P_1' \mid P_2'$, $P_2 = (\nu\, \mathrm{bn}(l_1) \cup \mathrm{bn}(l_2))(P_1'' \mid P_2'')$, $l = \tau$ και $P_1' \xrightarrow{l_1} P_1''$, $P_2' \xrightarrow{l_2} P_2''$, με $\mathrm{dual}(l_1, l_2)$. Το $\Gamma \vdash P_1' \mid P_2' \triangleright \Delta$ μπορεί να προκύψει μόνο από τον κανόνα (ParP), ο οποίος προϋποθέτει ότι $\Gamma \vdash P_1' \triangleright \Delta_1$ και $\Gamma \vdash P_2' \triangleright \Delta_2$. Χωρίς βλάβη της γενικότητας, υποθέτουμε πως $l_1 = x(y : T)$.

- Αν $l_2 = \overline{x}\langle y \rangle$, τότε, από την επαγωγική υπόθεση για το P_2, $\Gamma \vdash P_2'' \triangleright \Delta_2'$, με $\Delta_2' \lesssim \Delta_2$ και $y \in \mathrm{dom}(\Gamma)$. Επομένως, εφαρμόζοντας την επαγωγική υπόθεση στο P_1, παίρνουμε $\Gamma \vdash P_1'' \triangleright \Delta_1'$ με $\Delta_1' \lesssim \Delta_1$. Χρησιμοποιώντας τον κανόνα (ParP) του ελέγχου τύπων, καταλήγουμε πως $\Gamma \vdash P_2 \triangleright \Delta'$.
- Αν $l_2 = (\nu\, y : T)\overline{x}\langle y \rangle$, τότε, εφαρμόζοντας την επαγωγική υπόθεση, παίρνουμε $\Gamma_1 \vdash P_1'' \triangleright \Delta_1'$ και $\Gamma_2 \vdash P_2'' \triangleright \Delta_2'$, με $\Delta_1' \lesssim \Delta_1$, $\Delta_2' \lesssim \Delta_2$ και $\Gamma_1 \subseteq \Gamma \cdot y : T = \Gamma_2$. Αν $\Gamma_1 \neq \Gamma \cdot y : T$, τότε από το λήμμα 3.33 συνάγουμε $\Gamma \cdot y : T \vdash P_1'' \triangleright \Delta_1'$. Επομένως, ο κανόνας (ParP) του ελέγχου τύπων δίνει $\Gamma \cdot y : T \vdash P_1'' \mid P_2'' \triangleright \Delta'$ και ο κανόνας (ResNP) δίνει $\Gamma \vdash P_2 \triangleright \Delta'$.

Λόγω της παρατήρησης 3.26, $\Delta' \lesssim \Delta$.

(ParL) Έχουμε $P_1 = P \mid P''$ και $P_2 = P' \mid P''$. Από τον κανόνα (ParP) του ελέγχου τύπων, επειδή είναι ο μόνος που μπορεί να οδηγήσει στο $\Gamma \vdash P \mid P'' \triangleright \Delta$, ξέρουμε ότι $\Gamma \vdash P \triangleright \Delta_1$ και $\Gamma \vdash P'' \triangleright \Delta_2$, με $\Delta = \Delta_1 \uplus \Delta_2$. Από τη μετάβαση (ParL), ξέρουμε ότι $P \xrightarrow{l} P'$, το οποίο, από την επαγωγική υπόθεση και επειδή $\Gamma \vdash P \triangleright \Delta_1$, δίνει $\Gamma' \vdash P' \triangleright \Delta_1'$, με $\Delta_1' \lesssim \Delta_1$. Από το λήμμα 3.33, έχουμε $\Gamma' \vdash P'' \triangleright \Delta_2$. Εφαρμόζοντας ξανά τον κανόνα (ParP), παίρνουμε $\Gamma' \vdash P' \mid P'' \triangleright \Delta_1' \uplus \Delta_2$. Από την παρατήρηση 3.26, $\Delta_1' \uplus \Delta_2 \lesssim \Delta$.

(ParR) Τελείως όμοια με την προηγούμενη περίπτωση.

(ResN) Έχουμε $P_1 = (\nu\, x : T)P$ και $P_2 = (\nu\, x : T)P'$. Το $\Gamma \vdash P_1 \triangleright \Delta$ μπορεί να έχει προκύψει μόνο από τον κανόνα (ResNP), ο οποίος προϋποθέτει ότι $\Gamma \cdot x : T \vdash P \triangleright \Delta$. Από την επαγωγική υπόθεση, $\Gamma' \cdot x : T \vdash P' \triangleright \Delta'$, με $\Delta' \lesssim \Delta$. Εφαρμόζοντας τον κανόνα (ResNP), $\Gamma' \vdash (\nu\, x : T)P' \triangleright \Delta'$.

(Repl) Έχουμε $P_1 = !P$ και $P_2 = !P \mid P'$, με $P \xrightarrow{l} P'$. Το $\Gamma \vdash !P \triangleright \Delta$ μπορεί να προκύψει μόνο από τον κανόνα (Rep), ο οποίος προϋποθέτει ότι $\Gamma \vdash P \triangleright \Delta$. Από την επαγωγική υπόθεση, $\Gamma' \vdash P' \triangleright \Delta'$, με $\Delta' \lesssim \Delta$.

Από το $\Gamma \vdash !P \triangleright \Delta$ και το λήμμα 3.33 παίρνουμε $\Gamma' \vdash !P \triangleright \Delta$, άρα συνδυάζοντας τα $\Gamma' \vdash P' \triangleright \Delta'$ και $\Gamma' \vdash !P \triangleright \Delta$ με τον κανόνα (ParP) παίρνουμε $\Gamma' \vdash P_2 \triangleright \Delta \uplus \Delta'$. Επειδή $\Delta \uplus \Delta = \Delta$, η παρατήρηση 3.26 και η σχέση $\Delta' \lesssim \Delta$ συνεπάγονται $\Delta \uplus \Delta' \lesssim \Delta$.

(CondT) Έχουμε $P_1 = [x = x](P; P'')$, $P_2 = [\![x = x]\!]P'$ και $P \xrightarrow{l} P'$. Το $\Gamma \vdash [x = x](P; P'') \triangleright \Delta$ μπορεί να προκύψει μόνο από τον κανόνα (CondB), ο οποίος προϋποθέτει ότι $\Gamma \vdash [\![x = x]\!]P \triangleright \Delta_1$ και $\Gamma \vdash [\![x \neq x]\!]P'' \triangleright \Delta_2$, με $\Delta = \Delta_1 \uplus \Delta_2$, άρα $\Delta_1 \lesssim \Delta$. Από τη μετάβαση (CondX) του π-λογισμού, το $P \xrightarrow{l} P'$ συνεπάγεται $[\![x = x]\!]P \xrightarrow{l} [\![x = x]\!]P'$, άρα από την επαγωγική υπόθεση το $\Gamma \vdash [\![x = x]\!]P \triangleright \Delta_1$ δίνει $\Gamma' \vdash P_2 \triangleright \Delta'_1$, με $\Delta'_1 \lesssim \Delta_1 \lesssim \Delta$.

(CondF) Όμοια με την προηγούμενη περίπτωση.

(CondX) Έχουμε $P_1 = [\![x \ op \ y]\!]P$, $P_2 = [\![x \ op \ y]\!]P'$ και $P \xrightarrow{l} P'$. Το $\Gamma \vdash [\![x \ op \ y]\!]P \triangleright \Delta$ μπορεί να προκύψει μόνο από τον κανόνα (CondA), ο οποίος προϋποθέτει ότι $\Gamma \vdash x \triangleright X$ και $\Gamma \vdash P \triangleright \Delta_0$, με $\Delta = (X \ op \ y) \oplus \Delta_0$. Από την επαγωγική υπόθεση, $\Gamma' \vdash P' \triangleright \Delta'_0$, με $\Delta'_0 \lesssim \Delta_0$. Εφαρμόζοντας τον κανόνα (CondA), παίρνουμε $\Gamma' \vdash P_2 \triangleright \Delta'$, με $\Delta' = (X \ op \ y) \oplus \Delta'_0$ και από την παρατήρηση 3.26 $\Delta' \lesssim \Delta$.

(Alpha) Έχουμε $P_1 \equiv_\alpha P$ και $P \xrightarrow{l} P_2$. Από την πρόταση 3.30, το $\Gamma \vdash P_1 \triangleright \Delta$ δίνει $\Gamma \vdash P \triangleright \Delta$ και από την επαγωγική υπόθεση παίρνουμε κατ' ευθείαν το ζητούμενο.

2. Έστω $\Gamma \vdash S_1 \triangleright \Theta$ και $S_1 \xrightarrow{l} S_2$.

(ResGP) Έχουμε $S_1 = (\nu\, R)S$, $S_2 = (\nu\, R)S'$ και $S \xrightarrow{l} S'$. Το $\Gamma \vdash (\nu\, R)S \triangleright \Theta$ μπορεί να έχει προκύψει μόνο από τον κανόνα (ResGS) του ελέγχου τύπων, ο οποίος προϋποθέτει ότι $\Gamma \cdot R \vdash S \triangleright \Theta$. Από την επαγωγική υπόθεση, $\Gamma' \cdot R \vdash S' \triangleright \Theta'$, με $\Theta' \lesssim \Theta$. Εφαρμόζοντας τον κανόνα (ResGS) του ελέγχου τύπων, παίρνουμε $\Gamma' \vdash S_2 \triangleright \Theta'$.

(ResGS) Έχουμε $S_1 = (\nu\, G)P\,\langle u \rangle$, $S_2 = (\nu\, G)P'\,\langle u \rangle$ και $P \xrightarrow{l} P'$. Το $\Gamma \vdash (\nu\, G)P\,\langle u \rangle \triangleright \Theta$ μπορεί να έχει προκύψει μόνο από τον κανόνα (ResGS) του ελέγχου τύπων, ο οποίος προϋποθέτει ότι $\Gamma \vdash P \triangleright \Delta$, με $\Theta = G[u] \odot \Delta$. Από την περίπτωση των διεργασιών, $\Gamma' \vdash P' \triangleright \Delta'$, με $\Delta' \lesssim \Delta$. Εφαρμόζοντας τον κανόνα (ResGS) του ελέγχου τύπων, παίρνουμε $\Gamma' \vdash S_2 \triangleright \Theta'$, με $\Theta' = G[u] \odot \Delta'$. Από την παρατήρηση 3.26, έχουμε $\Theta' \lesssim \Theta$.

Οι υπόλοιπες περιπτώσεις αποδεικνύονται τελείως όμοια με τις αντίστοιχες περιπτώσεις για διεργασίες.

□

Συνέπεια 3.35. $\mathcal{P} \vdash_{\tilde{l}} S \ \wedge \ S \xrightarrow{l} S' \ \Rightarrow \ \mathcal{P} \vdash_{\tilde{l} \cup \{l\}} S'$

Απόδειξη. Εξ ορισμού, το $\mathcal{P} \vdash_{\tilde{l}} S$ σημαίνει πως $\Gamma_{\tilde{l}}(S) \vdash S \triangleright \Theta$ και $\mathcal{P} \vDash \Theta$. Από το προηγούμενο θεώρημα, $\Gamma' \vdash S' \triangleright \Theta'$, με $\Theta' \lesssim \Theta$ και το Γ' όπως ορίζεται στο θεώρημα. Επομένως, από την πρόταση 3.27, $\mathcal{P} \vDash \Theta'$ και λόγω του λήμματος 3.22, $\Gamma' = \Gamma_{\tilde{l} \cup \{l\}}(S')$. Καταλήγουμε λοιπόν πως $\mathcal{P} \vdash_{\tilde{l} \cup \{l\}} S'$. □

Συνέπεια 3.36. $\mathcal{P} \vdash S \ \wedge \ S \xrightarrow{l^*} S' \ \Rightarrow \ \neg\text{error}_{\mathcal{P},\tilde{l}}(S')$,
όπου $\tilde{l}$ *το σύνολο των μεταβάσεων που οδήγησαν από το* S *στο* S'.

Απόδειξη. Εφαρμόζοντας επαγωγικά τη συνέπεια 3.35, καταλήγουμε πως $\mathcal{P} \vdash_{\tilde{l}\cup\{l\}} S'$. Αυτό συνεπάγεται πως $\Gamma_{\tilde{l}}(S') \vdash S' \triangleright \Theta$ και $\mathcal{P} \vDash \Theta$. Λόγω του λήμματος 3.33, για κάθε περιβάλλον Γ Γ με $\Gamma_{\tilde{l}}(S') \subseteq \Gamma$ έχουμε $\Gamma \vdash S \triangleright \Theta$ και επομένως $\neg\text{error}_{\mathcal{P},\Gamma}(S')$. Οπότε, $\neg\text{error}_{\mathcal{P},\tilde{l}}(S')$. □

Κεφάλαιο 4

Η εκτελέσιμη υλοποίηση του ελέγχου τύπων σε Maude

Εδώ θα παρουσιάσουμε κώδικα σε Maude ο οποίος υλοποιεί τα όσα περιγράφησαν στο προηγούμενο κεφάλαιο. Οι τρεις πρώτες ενότητες βρίσκονται σε πλήρη αντιστοιχία με τις τρεις πρώτες ενότητες του προηγούμενου κεφαλαίου. Στην τελευταία ενότητα υλοποιούμε τα όσα αναφέρονται στο παράδειγμα 3.4 και τρέχουμε τον έλεγχο τύπων.

Συμβολισμός. Επειδή έχουμε χρησιμοποιήσει τον όρο «τύπος» για να αποδώσουμε τόσο το αγγλικό sort, που αντιστοιχεί στους τύπους της Maude, όσο και το αγγλικό type, που αντιστοιχεί στους τύπους του συστήματος ελέγχου τύπων που ορίσαμε, θα χρησιμοποιούμε κεφαλαίο Τ για να αναφερόμαστε στους Τύπους της Maude και πεζό για να αναφερόμαστε στους τύπους του ελέγχου τύπων.

Θα παρουσιαστούν τα βασικά σημεία του κώδικα, αφήνοντας κάποιες τεχνικές λεπτομέρειες για το παράρτημα Α΄· συγκεκριμένα, υποθέτουμε ότι όσα παρουσιάζονται παρακάτω βρίσκονται χωρίς πρόβλημα στο ίδιο άρθρωμα και ότι για οποιοδήποτε Τύπο `A` υπάρχουν οι Τύποι `Set{A}` και `NeSet{A}`, που αντιστοιχούν στα σύνολα στοιχείων του `A` και στα μη κενά σύνολα στοιχείων του `A` αντίστοιχα. Επιπλέον, για λόγους συντομίας της παρουσίασης, οι δηλώσεις μεταβλητών θα παρουσιάζονται στην αρχή κάθε ενότητας, με ισχύ για την ενότητα που παρουσιάστηκαν.

Η Maude ορίζει ότι `A` $<$ `NeSet{A}` $<$ `Set{A}` για τα σύνολα στοιχείων του `A` (άρα κάθε στοιχείο είναι ταυτόχρονα και μονοσύνολο) και τα εξοπλίζει με τους τελεστές `empty` για το κενό σύνολο, `_,_` για την παράθεση στοιχείων, `insert` για την εισαγωγή στοιχείου σε σύνολο, `delete` για τη διαγραφή στοιχείου από σύνολο, `_in_` για τον έλεγχο του ανήκειν, `|_|` για την πληθικότητα του συνόλου, `union` για την ένωση συνόλων, `intersection` για την τομή συνόλων, `_\_` για τη συνολοθεωρητική διαφορά, `_subset_` για τον έλεγχο αν ένα σύνολο είναι υποσύνολο ενός άλλου συνόλου και `_psubset_` για τον έλεγχο αν ένα σύνολο είναι γνήσιο υποσύνολο ενός άλλου συνόλου. Στα σύνολα δικαιωμάτων, αντί για τον τελεστή `_,_` θα χρησιμοποιήσουμε, για αισθητικούς κυρίως λόγους, απλή παράθεση των στοιχείων. Επειδή όταν οι Τύποι `A` και `B` βρίσκονται στο ίδιο είδος δημιουργούνται προβλήματα με τους ομώνυμους τελεστές, για τα σύνολα μεταβλητών πλαισίου θα χρησιμοποιήσουμε επί-

σης παράθεση αντί του _,_ και οι υπόλοιποι τελεστές θα έχουν επίθεμα -X (άρα ο delete γίνεται delete-X)· όμοια και για σύνολα τιμών μεταβλητών πλαισίου, με επίθεμα -V.

Η υλοποίηση χρησιμοποιεί την έκδοση 2.7 της (Core) Maude.

4.1 Η γλώσσα περιγραφής των πολιτικών ιδιωτικότητας

Οι μεταβλητές που θα χρησιμοποιήσουμε σε αυτή την ενότητα είναι οι ακόλουθες:

```
var BT : BasicType .
var G : Group .
var R : Role .
var USER : User .
vars Gs Gs' : Set{Group} .
var U : Purpose .
var US : Set{Purpose} .
vars P P1 P2 P' : Permission .
vars PS PS' PS1 PS2 : Set{Permission} .
vars H H1 H2 : Hierarchy .
vars HS HS' : Set{Hierarchy} .
vars POL POL1 POL2 : Policy .
vars PF PF1 PF2 : PermissionFunction .
vars COND COND' : Condition .
var CONDS : Set{Condition} .
```

Όπως σημειώνεται στον ορισμό 3.1 και στον ορισμό 3.7, η γλώσσα περιγραφής των πολιτικών ιδιωτικότητας προϋποθέτει κάποια σύνολα τα οποία καθορίζονται ανάλογα με την εκάστοτε εφαρμογή, κατά την υποστασιοποίηση. Σε αφηρημένο επίπεδο, το μόνο που μας ενδιαφέρει είναι η ύπαρξή τους· το μόνο που έχουμε να κάνουμε λοιπόν είναι να τα ορίσουμε ως Τύπους. Ορίζουμε επίσης τον Τύπο των ομάδων και δηλώνουμε ότι κάθε χρήστης και κάθε ρόλος αποτελούν ομάδες.

```
sorts BasicType User Role Group Purpose .
subsorts User Role < Group .
```

Θα συνεχίσουμε με τη γλώσσα έκφρασης προϋποθέσεων. Το προαπαιτούμενό της είναι ένα σύνολο μεταβλητών πλαισίου, η καθεμία από τις οποίες έχει ένα σύνολο τιμών. Χρησιμοποιούμε λοιπόν δύο Τύπους, έναν που αντιστοιχεί στις μεταβλητές πλαισίου και ένα που αντιστοιχεί στην ένωση των πεδίων τιμών τους· όπως και πιο πάνω, ο καθορισμός των συγκεκριμένων κάθε φορά μεταβλητών και τιμών εξαρτάται από την εκάστοτε εφαρμογή. Δηλώνουμε επιπλέον πως κάθε μεταβλητή πλαισίου θεωρείται βασικός τύπος. Ορίζουμε επίσης τον τελεστή domain, ο οποίος για κάθε μεταβλητή επιστρέφει τα στοιχεία του συνόλου τιμών της· υποθέτουμε πως αυτός καθορίζεται σε κάθε υποστασιοποίηση για κάθε μεταβλητή, με τον τρόπο που υποδεικνύεται στην ενότητα 4.4. Για λόγους συντομίας και επεκτασιμότητας, δηλώνουμε επίσης και τον Τύπο των τελεστών, που περιλαμβάνει τους τελεστές της ισότητας και της ανισότητας.

sorts CtxtVar CtxtVarValue CtxtVarOperator .
subsort CtxtVar < BasicType .
ops == =/= : –> CtxtVarOperator [**ctor**] .
op domain : CtxtVar –> Set{CtxtVarValue} .

Για τις προϋποθέσεις, έχουμε τα εξής:

sort Condition .
op ___ : CtxtVar CtxtVarOperator CtxtVarValue
–> [Condition] [**ctor prec** 14].
cmb X OP VAL : Condition **if** VAL V-in domain(X) .

op _/_ : Condition Condition –> Condition [**ctor assoc comm prec** 15].

Αφού δηλώσουμε ότι εφεξής θα υπάρχει ο Τύπος των προϋποθέσεων, δηλώνουμε ότι αν έχουμε μία μεταβλητή πλαισίου, έναν τελεστή και μία τιμή, τότε έχουμε μία προϋπόθεση –όχι πάντα καλοσχηματισμένη. Αμέσως μετά, δηλώνουμε ότι για να είναι όντως καλοσχηματισμένη η προϋπόθεση, πρέπει η τιμή να ανήκει στο πεδίο τιμών της μεταβλητής (ο τελεστής V-in αντιστοιχεί στη σχέση του ανήκειν στον Τύπο Set{CtxtVarValue}). Έπειτα δηλώνουμε ότι κάθε δύο προϋποθέσεις μπορούν να συνδυαστούν με τον τελεστή /\· οι λέξεις-κλειδιά **assoc** και **comm** δείχνουν ότι δεν έχει σημασία η σειρά των επιμέρους προϋποθέσεων. Ο χειροκίνητος ορισμός της ιδιότητας **prec** επιτρέπει την αποφυγή αμφισημιών όταν αργότερα θα έχουμε σύνθετες εκφράσεις για τις πολιτικές· ο τελεστής ___ δεν μπορεί ο ίδιος να προκαλέσει αμφισημία, αλλά ορίζοντας χειροκίνητα ένα χαμηλό **prec** για τον τελεστή /\ υποχρεούμαστε να κάνουμε το ίδιο είτε για την ιδιότητα **gather** του /\ είτε για την ιδιότητα **prec** του ___.

Πριν προχωρήσουμε στα υπόλοιπα στοιχεία της γλώσσας περιγραφής των πολιτικών ιδιωτικότητας, δηλώνουμε τα σχετικά με την διάταξη των προϋποθέσεων (ορισμός 3.3).

Αρχικά, χρειαζόμαστε τη συνάρτηση $\mathrm{vars}(\cdot)$. Η ομοιότητα της παρακάτω υλοποίησης με τη μαθηματική διατύπωση του ορισμού 3.2 είναι προφανής.

op vars : Condition –> Set{CtxtVar} .
eq vars(X OP VAL) = X .
eq vars(COND1 /\ COND2) = union-X(**vars**(COND1), **vars**(COND2)) .

Για τη διάταξη χρησιμοποιείται επίσης το θετικό πεδίο τιμών, το οποίο είναι καρτεσιανό γινόμενο άλλων συνόλων, κάτι που δεν αναπαρίσταται εύκολα στην Maude. Αντί λοιπόν να το χρησιμοποιήσουμε ως καρτεσιανό γινόμενο, θα εξετάσουμε ξεχωριστά τα επιμέρους σύνολα.

Παρατήρηση. Έστω $A_1, \ldots, A_n$ και $B_1, \ldots, B_n$ σύνολα. Αν $S = A_1 \times \ldots \times A_n$ και $B = B_1 \times \ldots \times B_n$, τότε $A \subseteq B \;\Leftrightarrow\; \forall i = 1, \ldots, n : A_i \subseteq B_i$.

Συνέπεια 4.1. $D^+(c_1{\restriction}_{\mathrm{vars}(c_2)}) \subseteq D^+(c_2) \;\Leftrightarrow\; \forall X \in \mathrm{vars}(c_2) : D^+(c_1{\restriction}_X) \subseteq D^+(c_2{\restriction}_X)$

Με βάση το παραπάνω, αρκεί να βρούμε τρόπο να υπολογίζουμε το $D^+(c{\restriction}_X)$ για αυθαίρετη προϋπόθεση και προβολές σε μία μόνο μεταβλητή πλαισίου κάθε φορά. Για να είναι πιο ξεκάθαρη η λειτουργία του, δίνουμε στον τελεστή $D^+(\cdot{\restriction}_\cdot)$ το όνομα allowed. Θεωρούμε επίσης ότι αν $X \notin \mathrm{vars}(c)$, τότε $\mathtt{allowed}(c, X) = D_X$.

```
1   op allowed : Condition CtxtVar -> Set{CtxtVarValue} .
2   eq allowed(X == VAL, Y) = if X == Y then VAL else domain(Y) fi .
3   eq allowed(X =/= VAL, Y) =
4       if X == Y then delete-V(VAL, domain(X)) else domain(Y) fi .
5   eq allowed(COND1 /\ COND2, X) =
6       intersection-V(allowed(COND1, X), allowed(COND2, X)) .
```

Ο τελεστής `allowed` δέχεται ως ορίσματα μία προϋπόθεση και μία μεταβλητή πλαισίου. Στις γραμμές 5 και 6 δηλώνεται ότι αν η προϋπόθεση δεν είναι ατομική, το θετικό πεδίο του περιορισμού της είναι η τομή των θετικών πεδίων του περιορισμού κάθε επιμέρους προϋπόθεσης. Στις γραμμές 2 έως 4 εξετάζεται η περίπτωση της ατομικής προϋπόθεσης: εδώ, ανάλογα με το αν η μεταβλητή πλαισίου που εμφανίζεται στην ατομική προϋπόθεση είναι η μεταβλητή πλαισίου που έχει δοθεί ως όρισμα, εφαρμόζουμε είτε τον ορισμό της σχέσης ικανοποίησης $\models$ (ορισμός 3.1) είτε την παραδοχή ότι $X \notin \mathrm{vars}(c)$ συνεπάγεται $\texttt{allowed}(c, X) = D_X$.

Είμαστε τώρα έτοιμοι για τον ορισμό της διάταξης προϋποθέσεων. Από τον ορισμό 3.3 και τη συνέπεια 4.1, έχουμε οτι

$$c_1 \leq c_2 \;\Leftrightarrow\; \mathrm{vars}(c_2) \subseteq \mathrm{vars}(c_1) \;\wedge\; \forall X \in \mathrm{vars}(c_2) : D^+(c_1{\restriction}_X) \subseteq D^+(c_2{\restriction}_X) \;,$$

γράφουμε λοιπόν

```
op _<=_ : Condition Condition -> Bool .
eq COND <= COND = true .
eq COND <= COND' = vars(COND') subset-X vars(COND)
    and $stricter-than(COND, COND', vars(COND')) .
op $stricter-than : Condition Condition Set{CtxtVar} -> Bool .
eq $stricter-than(COND, COND', empty-X) = true .
eq $stricter-than(COND, COND', (X XS)) =
    allowed(COND, X) subset-V allowed(COND', X)
    and $stricter-than(COND, COND', XS) .
```

όπου το $\forall X \in \mathrm{vars}(c_2)$ υλοποιείται μέσω της βοηθητικής συνάρτησης `$stricter-than` και η δήλωση **eq** `COND <=COND =true` δεν είναι απαραίτητη, αλλά χρησιμεύει για να συντομεύσει τη διαδικασία υπολογισμού στη συγκεκριμένη –πιθανώς συχνή– περίπτωση και επιπλέον δηλώνει ότι η σχέση $\leq$ έχει την ανακλαστική ιδιότητα.

Έχοντας ολοκληρώσει τα σχετικά με τις προϋποθέσεις, περνάμε στα δικαιώματα. Έχουμε δύο Τύπους δικαιωμάτων, τα δικαιώματα χωρίς προϋποθέσεις και τα δικαιώματα υπό προϋποθέσεις· έτσι, κατασκευάζουμε τους αντίστοιχους Τύπους στην Maude και ένα επιπλέον που τα περιλαμβάνει αμφότερα. Έπειτα, δηλώνουμε ότι κάθε δυνατό δικαίωμα χωρίς προϋπόθεση είναι της μορφής `read`, `write` `access` ή `disc G`, όπου το `G` είναι ομάδα, και ότι κάθε δυνατό δικαίωμα με προϋπόθεση είναι ένα δικαίωμα χωρίς προϋπόθεση ακολουθούμενο από **if** και μία προϋπόθεση. Ορίζουμε και την ιδιότητα **prec** ώστε ο τελεστής `_if_` να έχει μεγαλύτερη τιμή σε σχέση τόσο με τους τελεστές των προϋποθέσεων όσο και με τους τελεστές των δικαιωμάτων χωρίς προϋπόθεση.

```
sorts Permission UnconditionalPermission ConditionalPermission .
subsorts ConditionalPermission UnconditionalPermission <
    Permission .
```

```
ops read write access : -> UnconditionalPermission [ctor].
op disc_ : Group -> UnconditionalPermission [ctor prec 15].
op _if_ : UnconditionalPermission Condition
    -> ConditionalPermission [ctor prec 16] .
```

Ολοκληρώνοντας τα σχετικά με δικαιώματα, δηλώνουμε τα περί διάταξης δικαιωμάτων και συνόλων δικαιωμάτων. Επειδή στην υλοποίηση της Maude για τα σύνολα τα στοιχεία ενός Τύπου ταυτίζονται με τα μονοσύνολα του Τύπου, αρκεί να ορίσουμε τον τελεστή <= για σύνολα δικαιωμάτων. Κάνουμε επαγωγή στο πρώτο όρισμα (το οποίο μπορεί να είναι είτε το κενό σύνολο είτε ένα στοιχείο μαζί με ένα σύνολο) και χρησιμοποιούμε τον ορισμό 3.3:

```
op _<=_ : Set{Permission} Set{Permission} -> Bool .
eq empty <= PS = true .
eq P if COND PS <= P if COND' PS' =
    COND <= COND' and PS <= P if COND' PS' .
eq P if COND PS <= P PS' = PS <= P PS' .
eq P PS <= P PS' = PS <= P PS' .
eq PS1 <= PS2 = false [owise] .
```

Πράγματι, από τον ορισμό του, το πρώτο όρισμα μπορεί να είναι είτε το κενό σύνολο, το οποίο εξ ορισμού είναι αυστηρότερο από κάθε άλλο σύνολο, είτε ένα σύνολο με τουλάχιστον ένα δικαίωμα. Στη δεύτερη περίπτωση, απαιτείται στο δεύτερο όρισμα να υπάρχει ένα δικαίωμα λιγότερο αυστηρό· αυτό μπορεί να γίνει με τρεις τρόπους. Το προηγούμενο πρέπει να συμβαίνει για όλα τα στοιχεία του πρώτου ορίσματος· αν οποιαδήποτε στιγμή δεν μπορούμε να συνεχίσουμε, τότε το πρώτο όρισμα δεν είναι αυστηρότερο του δεύτερου.

Για τις συναρτήσεις απόδοσης δικαιωμάτων, χρησιμοποιούμε σύνταξη παρόμοια με το μαθηματικό συμβολισμό που δώσαμε στο προηγούμενο κεφάλαιο. Επειδή οι τελεστές περιέχουν ειδικούς χαρακτήρες, χρησιμοποιούμε τη μορφή μοναδικού αναγνωριστικού. Δηλώνουμε λοιπόν ότι κάθε συνάρτηση απόδοσης δικαιωμάτων είναι είτε μία απόδοση ενός συνόλου δικαιωμάτων σε μία ομάδα για ένα σκοπό είτε η παράθεση (η σειρά δεν έχει σημασία) δύο επιμέρους συναρτήσεων απόδοσης δικαιωμάτων. Ορίζουμε επίσης έναν τελεστή ο οποίος με δεδομένα μία συνάρτηση απόδοσης δικαιωμάτων, έναν σκοπό και μία ομάδα επιστρέφει το σύνολο δικαιωμάτων που αποδίδεται στην εν λόγω ομάδα για τον εν λόγω σκοπό από την εν λόγω συνάρτηση απόδοσης δικαιωμάτων.

```
sort PermissionFunction .
op (p`(_`,_`)=_) : Purpose Group Set{Permission}
    -> PermissionFunction [ctor prec 29] .
op _`,_ : PermissionFunction PermissionFunction
    -> PermissionFunction [ctor assoc comm prec 30] .

op (_`(_`,_`)) : PermissionFunction Purpose Group
    -> Set{Permission} .
eq (p(U, G) = PS)(U, G) = PS .
eq (PF1, PF2)(U,G) = union(PF1(U,G), PF2(U,G)) .
eq PF(U,G) = empty [owise] .
```

Για τις ιεραρχίες, έχουμε τα εξής:

```
sort Hierarchy
op nil : -> Hierarchy [ctor].
op _:_‘[_‘] : Group Set{Purpose} NeSet{Hierarchy} ~> Hierarchy
    [ctor].

op _:_ : Group Set{Purpose} -> Hierarchy .
eq G : US = G : US[nil] .
op _‘[_‘] : Group NeSet{Hierarchy} ~> Hierarchy .
eq G[HS] = G : empty[HS] .
op _‘[‘] : Group -> Hierarchy .
eq G []  = G : empty[nil] .

op cyclicalHierarchyError : -> [Hierarchy] .

cmb R : US[HS] : Hierarchy if not R in groups(HS) .
cmb USER : US[HS] : Hierarchy if HS == nil .
ceq G : US[HS] = cyclicalHierarchyError if G in groups(HS) .

op groups : Set{Hierarchy} -> Set{Group} .
eq groups(nil) = empty .
eq groups(G : US[HS]) = insert(G, groups(HS)).
eq groups(empty) = empty .
eq groups((H, HS)) = union(groups(H), groups(HS)) .

op root : Hierarchy -> Set{Group} .
eq root(nil) = empty .
eq root(G : US[HS]) = G .

op _->_ : Purpose Set{Hierarchy} -> Set{Hierarchy} .
eq U -> nil = nil .
eq U -> G : US[HS] = G : insert(U, US)[HS] .
eq U -> (H, HS) = (U -> H, U -> HS) .
```

Αρχικά, ορίζουμε τον αντίστοιχο Τύπο και δηλώνουμε ότι κάθε (καλοσχηματισμένη) ιεραρχία είναι είτε η κενή ιεραρχία είτε της μορφής $G : \tilde{u}[\tilde{H}]$, με το $\tilde{H}$ μη κενό (ο τελεστής _:_[_] δηλώνεται σε μορφή μοναδικού αναγνωριστικού)· η δεύτερη μορφή γενικεύει τις μορφές $R : \tilde{u}[\tilde{H}]$ (με $\tilde{H}$ μη κενό) και $U : \tilde{u}[\epsilon]$ που ορίστηκαν στον ορισμό 3.1, αφήνοντας όμως περιθώριο για περισσότερες (κακοσχηματισμένες σύμφωνα με τον ορισμό 3.1) ιεραρχίες. Δηλώνουμε λοιπόν ότι μία ιεραρχία της μορφής $R : \tilde{u}[\tilde{H}]$ είναι καλοσχηματισμένη αν και μόνο αν $R \notin \text{groups}(\tilde{H})$ και ότι μία ιεραρχία της μορφής $R : \tilde{u}[\tilde{H}]$ είναι καλοσχηματισμένη αν και μόνο αν $\tilde{H} = \{\epsilon\}$. Έπειτα, υλοποιούμε τους συμβολισμούς που εισάγαμε στη σελίδα 24: για τον καθένα από τους τρεις συμβολισμούς δηλώνουμε έναν τελεστή και έναν κανόνα στην εξισωτική λογική που δείχνει ποια ιεραρχία συμβολίζεται. Ορίζουμε επίσης τον τελεστή cyclicalHierarchyError, ο οποίος έχει είδος αλλά όχι Τύπο –είναι δηλαδή ένα σφάλμα– και χρησιμεύει στην αποσφαλμάτωση κακοσχηματισμένων ιεραρχιών Τέλος, ορίζουμε τον τελεστή groups, ο οποίος δοθείσης μίας ιεραρχίας ή ενός συνόλου ιεραρχιών (όπως και πιο πάνω, εκμεταλλευόμαστε το γεγονός ότι στην Maude τα στοιχεία ενός Τύπου ταυ-

τίζονται με τα μονοσύνολα των συνόλων του Τύπου) συλλέγει τις ομάδες που εμφανίζονται σε αυτή, τον τελεστή root, που επιστρέφει τη ρίζα μιας ιεραρχίας, και τον τελεστή _−>_ που προσθέτει έναν σκοπό στη ρίζα μιας ιεραρχίας· θα διευκολυνθούμε παρακάτω αν χρησιμοποιήσουμε τον τελεστή _−>_ σε σύνολα ιεραρχιών, οπότε τον επεκτείνουμε (με φυσικό τρόπο).

Για τις πολιτικές ιδιωτικότητας, ορίζουμε τον αντίστοιχο Τύπο και δηλώνουμε ότι κάθε (καλοσχηματισμένη) πολιτική ιδιωτικότητας είναι είτε της μορφής $t \gg H, pf$ (ο τελεστής δηλώνεται σε μορφή μοναδικού αναγνωριστικού) είτε μία παράθεση, χωρίς να έχει σημασία η σειρά, επιμέρους (καλοσχηματισμένων) πολιτικών ιδιωτικότητας. Ορίζουμε, με τετριμμένο τρόπο, έναν τελεστή types, ο οποίος για κάθε πολιτική επιστρέφει το σύνολο των βασικών τύπων που εμφανίζονται σε αυτήν, και δηλώνουμε ότι μία παράθεση δύο (και επομένως –επαγωγικά– περισσότερων) πολιτικών είναι καλοσχηματισμένη αν δεν εμφανίζεται ο ίδιος τύπος στις επιμέρους πολιτικές. Όπως και πιο πάνω, χρησιμοποιούμε τον τελεστή multiplePoliciesForType ως αντικείμενο σφάλματος για ευκολότερη αποσφαλμάτωση σε περίπτωση που έχουμε κακοσχηματισμένη πολιτική.

```
sort Policy .
op _>>_‘,_ : BasicType Hierarchy PermissionFunction
    −> Policy [ctor prec 35] .
op _;_ : Policy Policy ~> Policy [ctor assoc comm prec 40] .

op multiplePoliciesForType : Type −> [Policy] .

op types : Policy −> Set{Type} .
eq types(BT >> H , PF) = BT .
eq types(POL1 ; POL2) = union(types(POL1), types(POL2)) .

cmb POL1 ; POL2 : Policy
    if intersection(types(POL1), types(POL2)) = empty .
eq BT >> H1, PF1 ; BT >> H2, PF2 = multiplePoliciesForType(BT) .
```

4.2 Ο π-λογισμός

Ο έλεγχος τύπων γίνεται στατικά, επομένως χρειαζόμαστε τα προγράμματα του π-λογισμού ως συντακτικές οντότητες. Αυτό συνεπάγεται πως δεν χρειάζεται για τους σκοπούς μας να υλοποιήσουμε τη σημασιολογία.

Σημείωση. Σε περίπτωση που υλοποιούσαμε τη σημασιολογία, ο μη ντετερμινισμός των μεταβάσεων θα μας οδηγούσε στη χρήση κανόνων αναγραφής αντί για εξισωτικούς κανόνες. Δυστυχώς, οι κανόνες αναγραφής της Maude δεν υποστηρίζουν ετικέτες με μεταβλητές, όπως αυτές που χρειαζόμαστε εδώ. Ταυτόχρονα, το ότι κατά τη λήψη μηνυμάτων, τα ονόματα που είναι δυνατόν να ληφθούν είναι άπειρα, με τη συμπεριφορά του συστήματος να είναι εν γένει διαφορετική για το καθένα, καθώς και το ότι κατά την ομοιόμορφη αντικατάσταση ονομάτων ενδέχεται να απαιτούνταν η μετονομασία ενός δεσμευμένου ονόματος και τα διαθέσιμα ονόματα είναι άπειρα θα οδηγούσαν –σε μια αφελή υλοποίηση– σε έκρηξη καταστάσεων. Το [35] δίνει λύσεις για αυτά τα ζητήματα: για το πρώτο, θεωρεί την ετικέτα μέρος της

μετάβασης και ορίζει τις μεταβάσεις κατά τρόπο που να «αποθηκεύουν» τις ετικέτες· για το δεύτερο, συνοδεύει κάθε εν γένει διαφορετικό όνομα με έναν φυσικό αριθμό που το ξεχωρίζει από τα υπόλοιπα και θεωρεί ότι οι μεταβάσεις γίνονται σε ένα πεπερασμένο περιβάλλον ονομάτων, το οποίο προκύπτει από τις διεργασίες με τις οποίες μπορεί να ανταλλάξει μηνύματα.

Οι μεταβλητές που θα χρησιμοποιήσουμε σε αυτή την ενότητα είναι οι ακόλουθες:

```
vars X Y : Name .
var T : Type .
vars P P1 P2 : Process .
vars NEP1 NEP2 : NeProcess .
var S : System .
vars NES1 NES2 : NeSystem .
var OP : CtxtVarOperator .
var G : Group .
var R : Role .
var U : Purpose .
```

Σημειώνεται ότι σε σχέση με την προηγούμενη ενότητα αλλάζουν σημασία τα σύμβολα `X`, `Y`, `XS`, `P`.

Ακολουθώντας τον ορισμό 3.8, αρχικά δηλώνουμε την ύπαρξη του συνόλου των ονομάτων και περιλαμβάνουμε σε αυτά τις τιμές των μεταβλητών πλαισίου.

```
sort Name .
subsort CtxtVarValue < Name .
```

Έπειτα, δηλώνουμε ότι οι βασικοί τύποι, καθώς και κάθε αντικείμενο της μορφής $G[T]$, με G ομάδα και T τύπο, αποτελούν τύπους.

```
sort Type .
subsort BasicType < Type .
op _‘[_‘] : Group Type -> Type [prec 15] .
```

Μπορούμε τώρα να ορίσουμε τις διεργασίες:

```
1   sorts Process NeProcess .
2   subsort NeProcess < Process .

4   op 0P : -> Process [ctor] .

6   op !_ : Process -> NeProcess [frozen ctor prec 17] .

8   op _|_ : Process Process -> Process
9       [frozen ctor assoc comm id: 0P prec 19] .
10  op _|_ : Process NeProcess -> NeProcess [ditto] .

12  op (‘(v_:_‘)_) : Name Type Process ~> NeProcess
13      [frozen(3) ctor prec 17] .
14  cmb (v X : T) P : NeProcess if not X :: CtxtVarValue .

16  op (_‘(_:_‘)._) : Name Name Type Process ~> NeProcess
```

```
17          [frozen(4) ctor prec 17] .
18      cmb X(Y : T). P : NeProcess if not Y :: CtxtVarValue .

20      op _<_>._ : Name Name Process -> NeProcess [frozen(3) ctor prec 17].

22      op (`[_==_`]`(_;_`)) : Name CtxtVarValue Process Process
23          -> NeProcess [frozen(3 4) ctor] .

25      op `[`[___`]`]_ : Name CtxtVarOperator CtxtVarValue Process
26          -> NeProcess [frozen(4) ctor prec 17] .

28      op `[___`]_ : Name CtxtVarOperator CtxtVarValue Process
29          -> NeProcess [frozen(4) prec 17] .
30      eq [X == Y]P = [X == Y] (P ; 0P) .
31      eq [X =/= Y]P = [X == Y] (0P ; P) .
```

Κατ' αρχάς, παρατηρούμε ότι σε όλους τους τελεστές έχουμε ορίσει την ιδιότητα **frozen** για τα ορίσματα που αποτελούν διεργασίες. Αυτό συμβαίνει διότι διαφορετικά, όταν κάποια στιγμή στο μέλλον χρειαστεί να ορίσουμε τη σημασιολογία των προγραμμάτων, η Maude θα ψάχνει σε όλο το συντακτικό δέντρο μίας διεργασίας για όρους που μπορούν να αναγραφούν· η σημασιολογία των μεταβάσεων, όμως, δεν προβλέπει μετάβαση που δεν περιλαμβάνει τη ρίζα του συντακτικού δέντρου. Χρησιμοποιούμε δύο Τύπους, έναν για όλες τις διεργασείς και έναν για όλα τις διεργασίες πλην της κενής. Αρχικά, δηλώνουμε την ύπαρξη του κενού συστήματος. Στις γραμμές 8 έως 10 δηλώνουμε τα σχετικά με την παράλληλη σύνθεση διεργασιών: στη γραμμή 9 έχουμε δηλώσει –αν και δεν ήταν απαραίτητο για τους σκοπούς μας– ότι η παράλληλη σύνθεση διεργασιών έχει την αντιμεταθετική και την προσεταιριστική ιδιότητα και ουδέτερο στοιχείο το κενό σύστημα (χρησιμοποιούμε έτσι ένα υποσύνολο της ομοιότητας συστημάτων, όπως ορίστηκε στον ορισμό 3.11)· στη γραμμή 10 επαναλαμβάνουμε τον ορισμό για να καθορίσουμε τη συμπεριφορά του τελεστή σε σχέση με τον υποτύπο NeProcess –η ιδιότητα **ditto** διατηρεί τις ιδιότητες που έχουμε δώσει στη γενική περίπτωση. Στις γραμμές 12 έως 14 δηλώνουμε τα σχετικά με τη δέσμευση ονομάτων σε διεργασίες: το σύμβολο ~> στη γραμμή 12 δηλώνει ότι δεν είναι κάθε δέσμευση επιτρεπτή· η γραμμή 14 ορίζει πως επιτρεπτές δεσμεύσεις είναι αυτές όπου το όνομα δεν είναι τιμή μεταβλητής πλαισίου. Τα ίδια ισχύουν και για τον ορισμό της λήψης μηνυμάτων στις γραμμές 16 έως 18. Έπειτα, ορίζουμε τα στοιχεία του ελέγχου υποθέσεων, δηλαδή τις διεργασίες της μορφής $[x = y](P_1; P_2)$ και $[\![x\ op\ y]\!]P$, καθώς και τις συντομεύσεις (εξ ου και δεν έχουν την ιδιότητα **ctor**) $[x = y]P$ και $[x \neq y]P$.

Για τα συστήματα, με το ίδιο σκεπτικό που έχουμε παρουσιάσει για τις διεργασίες, γράφουμε:

```
1       sorts System NeSystem .
2       subsort NeSystem < System .

4       op 0S : -> System [ctor] .

6       op _|_ : System System -> System
7           [frozen ctor assoc comm id: 0S prec 25] .
```

```
8     op _|_ : System NeSystem -> NeSystem [ditto] .

10    op (`(v_:_`)_) : Name Type System ~> NeSystem
11        [frozen(3) ctor prec 23].
12    cmb (v X : T) S : NeSystem if not X :: CtxtVarValue .

14    op (`(v_`)_<_>) : Group Process Purpose -> NeSystem
15        [frozen(2) ctor prec 21] .

17    op (`(v_`)_) : Role System -> NeSystem [frozen(2) ctor prec 23] .
```

Τα όσα αφορούν ελεύθερα και δεσμευμένα ονόματα και ομάδες ακολουθούν το σχήμα 3.3. Παρουσιάζουμε ενδεικτικά τα περί ελεύθερων ονομάτων διεργασιών (τα υπόλοιπα μπορούν να βρεθούν στο παράρτημα Α΄):

```
op fn : Process -> Set{Name} .
eq fn(0P) = empty .
eq fn(! P) = fn(P) .
eq fn(NEP1 | NEP2) = union(fn(NEP1), fn(NEP2)) .
eq fn((v X : T) P) = delete(X, fn(P)) .
eq fn(X (Y : T). P) = insert(X, delete(Y, fn(P))) .
eq fn(X < Y >. P) = insert(X, insert(Y, fn(P))) .
eq fn([X == Y] (P1 ; P2)) =
    insert(X, insert(Y, union(fn(P1), fn(P2)))) .
eq fn([[X OP Y]] P) = insert(X, insert(Y, fn(P))) .
```

4.3 Το σύστημα ελέγχου τύπων

Οι μεταβλητές που θα χρησιμοποιήσουμε σε αυτή την ενότητα είναι οι ακόλουθες:

```
vars X Y : Name .
vars P P' P1 P2 : Process .
vars NEP1 NEP2 : NeProcess .
vars S S1 S2 : System .
vars NES1 NES2 : NeSystem .

var OP : CtxtVarOperator .
var BT : BasicType .
vars T T' : Type .
var TS : Set{Type} .
var R : Role .
vars G G' : Group .
var GS : Set{Group} .
var U : Purpose .
var US : Set{Purpose} .
vars PS PS1 PS2 : Set{Permission} .
vars COND COND' : Condition .
var CONDS : Set{Condition} .
var H : Hierarchy .
```

```
vars HS HS' : Set{Hierarchy} .
var POL : Policy .
var PF : PermissionFunction .

vars D D1 D2 : DEnvironment .
vars NED NED1 NED2 : NeDEnvironment .
vars GAMMA GAMMA1 GAMMA2 : GEnvironment .
vars TH TH1 TH2 : ThInterface .
vars NETH1 NETH2 : NeThInterface .
var HTH : InterfaceHierarchy .
```

Ορίζουμε τώρα τα περιβάλλοντα Γ:

```
sorts GEnvironment .
op none-G : -> GEnvironment [ctor] .
op _:_ : Name Type -> GEnvironment [ctor prec 35] .
op group:_ : Group -> GEnvironment [ctor prec 35] .
op _._ : GEnvironment GEnvironment -> GEnvironment
    [ctor assoc comm id: none-G prec 40] .

op nameAppearsTwice : Name -> [GEnvironment] .
eq X : T . X : T' = nameAppearsTwice(X) .
op groupAppearsTwice : Group -> [GEnvironment] .
eq group: G . group: G = groupAppearsTwice(G) .

op groups : GEnvironment -> Set{Group} .
eq groups(none-G) = empty .
eq groups(X : T) = empty .
eq groups(group: G) = G .
eq groups(GAMMA1 . GAMMA2) = union(groups(GAMMA1), groups(GAMMA2)).
```

Αρχικά δηλώνουμε ότι υπάρχει ο Τύπος των περιβαλλόντων Γ και πως κάθε περιβάλλον Γ είναι είτε το κενό είτε η αντιστοίχιση ενός ονόματος σε έναν τύπο είτε η καταγραφή μιας ομάδας είτε η (προσεταιριστική, αντιμεταθετική και με ουδέτερο στοιχείο το κενό περιβάλλον) παράθεση δύο επιμέρους περιβαλλόντων. Παρατηρούμε ότι για την καταγραφή μιας ομάδας χρησιμοποιήσαμε το πρόθεμα group:, σε αντιδιαστολή με τη μαθηματική περιγραφή, όπου απλά παραθέταμε την ίδια την ομάδα· αυτή η επιλογή έγινε επειδή διαφορετικά μία συμβολοσειρά της μορφής G, όπου G κάποια ομάδα, θα ανήκε ταυτόχρονα στους Τύπους των ομάδων και των περιβαλλόντων Γ, τα οποία όμως δεν υπάρχει διαισθητικός λόγος να βρίσκονται στο ίδιο είδος. Έπειτα, καταγράφουμε τον περιορισμό πως κάθε όνομα και κάθε ομάδα πρέπει να εμφανίζονται σε ένα περιβάλλον Γ το πολύ μία φορά, ως εξής: δημιουργούμε δύο τελεστές «σφάλματος», οι οποίοι αντικαθιστούν τις προβληματικές καταχωρήσεις και επομένως κάνουν το περιβάλλον Γ μη λειτουργικό. Τέλος, κάνοντας επαγωγή στη δομή του περιβάλλοντος Γ, ορίζουμε τον τελεστή groups που επιστρέφει τις ομάδες που εμφανίζονται στο περιβάλλον Γ.

Για τα περιβάλλοντα Δ, γράφουμε:

```
sorts DEnvironment NeDEnvironment .
subsort NeDEnvironment < DEnvironment .
op none-D : -> DEnvironment [ctor] .
```

```
op _:_ : BasicType Set{Permission} -> NeDEnvironment
    [ctor prec 35] .
op _._ : DEnvironment DEnvironment ~> DEnvironment
    [ctor assoc comm id: none-D prec 40] .
op _._ : NeDEnvironment DEnvironment ~> NeDEnvironment
    [ctor assoc comm id: none-D prec 40] .
cmb NED1 . NED2 : NeDEnvironment
    if intersection(types(NED1), types(NED2)) = empty .

op types : DEnvironment -> Set{Type} .
eq types(none-D) = empty .
eq types(BT : PS) = BT .
eq types(D1 . D2) = union(types(D1), types(D2)) .

op permissions : DEnvironment BasicType -> Set{Permission} .
eq permissions(BT : PS . D, BT) = PS .
eq permissions(D, BT) = empty [owise] .
```

Χρησιμοποιούμε έναν Τύπο για όλα τα περιβάλλοντα Δ και ένα για τα μη κενά. Δηλώνουμε πως κάθε περιβάλλον Δ είτε είναι το κενό είτε αντιστοιχίζει έναν βασικό τύπο με ένα σύνολο δικαιωμάτων (οπότε είναι μη κενό) είτε είναι παράθεση επιμέρους περιβαλλόντων Δ· η παράθεση είναι προσεταιριστική και αντιμεταθετική με ουδέτερο στοιχείο το κενό περιβάλλον, οδηγεί σε μη κενό περιβάλλον αν ένα από τα επιμέρους είναι μη κενό και, τέλος, ορίζεται καλώς μόνο αν ο ίδιος βασικός τύπος δεν εμφανίζεται δύο φορές. Ορίζουμε επίσης τους τελεστές types, που επιστρέφει τους τύπους που εμφανίζονται σε ένα περιβάλλον Δ και permissions, που επιστρέφει το σύνολο δικαιωμάτων που αντιστοιχίζεται με έναν δεδομένο βασικό τύπο στο δεδομένο περιβάλλον Δ· ο ορισμός του πρώτου κάνει επαγωγή στη δομή του περιβάλλοντος, ενώ ο ορισμός του δεύτερου χρησιμοποιεί τις αλγεβρικές ιδιότητες της παράθεσης, καθώς και το ότι κάθε βασικός τύπος εμφανίζεται το πολύ μία φορά.

Μαζί με τις διεπαφές Θ, δηλώνουμε και τα σχετικά με ιεραρχίες διεπαφής:

```
sorts InterfaceHierarchy ThInterface NeThInterface .
subsort NeThInterface < ThInterface .
subsort InterfaceHierarchy < Hierarchy .

op _`[_`] : Group Purpose -> InterfaceHierarchy [ctor] .
op _`[_`] : Group InterfaceHierarchy -> InterfaceHierarchy
    [ctor].
eq G[US] = G : US [nil] .
cmb G : US [HS] : InterfaceHierarchy
    if (US == empty and | HS | == 1 and HS =/= nil)
    or (HS == nil and | US | == 1) .

op purpose : InterfaceHierarchy -> Purpose .
eq purpose(G : U [nil]) = U .
eq purpose(G : empty [H]) = purpose(H) .
```

```
op none-Th : -> ThInterface [ctor] .
op _>>'<_',_> : BasicType InterfaceHierarchy Set{Permission}
    -> NeThInterface [ctor prec 35] .
op _;_ : ThInterface ThInterface -> ThInterface
    [ctor assoc comm id: none-Th prec 40] .
op _;_ : NeThInterface ThInterface -> NeThInterface [ditto] .
```

Όπως και στα προηγούμενα χρησιμοποιούμε δύο Τύπους για να περιγράψουμε τις διεπαφές Θ. Δηλώνουμε επίσης πως οι ιεραρχίες διεπαφής είναι ειδική περίπτωση ιεραρχιών, οι οποίες κατασκευάζονται είτε περικλείοντας ένα σκοπό u με μία ομάδα G (το οποίο είναι ισοδύναμο με την ιεραρχία $G : u[\epsilon]$) είτε περικλείοντας μια ιεραρχία διεπαφής με μια ομάδα (για τον ορισμό των τελεστών, χρησιμοποιούμε μορφή μοναδικού αναγνωριστικού). Καθορίζουμε επιπλέον πως αν $G : \tilde{u}[\tilde{H}]$ είναι μια ιεραρχία, τότε για να είναι και ιεραρχία διεπαφής πρέπει είτε να μην υπάρχει σκοπός και το $\tilde{H}$ να περιέχει μόνο μία μη κενή ιεραρχία είτε να υπάρχει μοναδικός σκοπός και το $\tilde{H}$ να περιέχει μόνο την κενή ιεραρχία. Μπορούμε τώρα να δηλώσουμε τις διεπαφές Θ, με σκεπτικό παρόμοιο με εκείνο που έχουμε χρησιμοποιήσει για τα περιβάλλοντα Δ· να σημειωθεί μόνο πως ο τελεστής _>>'<_',_> είναι σε μορφή μοναδικού αναγνωριστικού, με το πρώτο ' να αντιστοιχεί σε χαρακτήρα κενού και το δεύτερο να χρησιμοποιείται λόγω του , που ακολουθεί.

Δηλώνουμε τώρα τις βοηθητικές συναρτήσεις του ορισμού 3.14. Για όλες τις πράξεις, χρησιμοποιούμε το σύμβολο + αντί των $\uplus$, $\odot$ και $\oplus$. Για τις περισσότερες, ο ορισμός τους στην Maude είναι απλή μεταγραφή του μαθηματικού ορισμού:

```
op _+_ : Condition Set{Permission} -> Set{Permission} [prec 17] .
eq COND + P+:UnconditionalPermission =
    P+:UnconditionalPermission if COND .
eq COND + P if COND' = P if COND /\ COND' .
eq COND + empty = empty .
eq COND + (P PS) = (COND + P) (COND + PS) .

op _+_ : Condition DEnvironment -> DEnvironment .
eq COND + none-D = none-D .
eq COND + BT : PS = BT : COND + PS .
eq COND + (D1 . D2) = (COND + D1) . (COND + D2) .

op Dr : Type -> DEnvironment .
eq Dr(BT) = BT : read .
eq Dr(G [BT]) = BT : access .
eq Dr(T) = none-D [owise] .

op Dw : Type -> DEnvironment .
eq Dw(G[BT]) = BT : write .
eq Dw(G [G' [BT]]) = BT : disc G .
eq Dw(T) = none-D [owise] .

op _'[_']+_ : Group Purpose DEnvironment -> ThInterface [prec 35] .
op _'[_']+_ : Group Purpose NeDEnvironment
```

```
    -> NeThInterface [ditto].
  eq G [ U ] + none-D = none-Th .
  eq G [ U ] + (BT : PS) = BT >> < G [U] , PS > .
  eq G [ U ] + (D1 . D2) = G [ U ] + D1 ; G [ U ] + D2 .

  op _+_ : Group ThInterface -> ThInterface [prec 35] .
  op _+_ : Group NeThInterface -> NeThInterface [ditto] .
  eq G + none-Th = none-Th .
  eq G + (BT >> < HTH, PS >) = BT >> < G [HTH], PS > .
  eq G + (TH1 ; TH2) = G + TH1 ; G + TH2 .
```

Η μόνη περίπτωση που θα εξετάσουμε αναλυτικά είναι η $\Delta_1 \uplus \Delta_2$. Κατ' αρχάς, παρατηρούμε ότι η πράξη είναι εξ ορισμού προσεταιριστική και αντιμεταθετική, με ουδέτερο στοιχείο το κενό περιβάλλον Δ και, επιπλέον, ότι αν ένα όρισμα είναι μη κενό, τότε και το αποτέλεσμα είναι μη κενό. Για την υλοποίηση, συλλέγουμε πρώτα όλους τους τύπους που εμφανίζονται στα Δ_1 και Δ_2 και έπειτα, χρησιμοποιώντας τη βοηθητική συνάρτηση $+, για κάθε τύπο προσθέτουμε στο αποτέλεσμα μια αντιστοίχισή του με την ένωση των δικαιωμάτων που του αντιστοιχίζονται στα Δ_1 και Δ_2.

```
  op _+_ : DEnvironment DEnvironment -> DEnvironment
      [assoc comm id: none-D prec 41] .
  op _+_ : NeDEnvironment DEnvironment -> NeDEnvironment [ditto].

  eq NED1 + NED2 =
      $+(NED1, NED2, union(types(NED1), types(NED2))) .

  op $+ : NeDEnvironment NeDEnvironment Set{Type}
      -> NeDEnvironment .
  eq $+(D1, D2, empty) = none-D .
  eq $+(D1, D2, (BT, TS)) = $+(D1, D2, TS) .
      BT : union(permissions(D1, BT), permissions(D2,BT)) .
```

Είμαστε τώρα έτοιμοι να ορίσουμε το σύστημα ελέγχου τύπων. Για να το κάνουμε, θα εκμεταλλευτούμε την παρακάτω πρόταση:

Πρόταση 4.2. *Έστω περιβάλλον Γ Γ, όνομα x, διεργασία P και σύστημα S.*

1. *Αν υπάρχει τύπος T ώστε $\Gamma \vdash x \triangleright T$, τότε είναι μοναδικός.*
2. *Αν υπάρχει περιβάλλον Δ Δ ώστε $\Gamma \vdash P \triangleright \Delta$, τότε είναι μοναδικό.*
3. *Αν υπάρχει διεπαφή Θ Θ ώστε $\Gamma \vdash S \triangleright \Theta$, τότε είναι μοναδική.*

Απόδειξη.

1. Για να έχουμε ότι $\Gamma \vdash x \triangleright T$ και $\Gamma \vdash x \triangleright T'$ με $T \neq T'$, πρέπει το Γ να είναι της μορφής $\Gamma = \Gamma' \cdot x : T \cdot x : T'$, κάτι που απαγορεύεται από τον ορισμό των περιβαλλόντων Γ.
2. Θα αποδείξουμε την περίπτωση αυτή με επαγωγή στη μορφή της διεργασίας. Έστω ότι υπάρχουν Γ, P, Δ_1 και Δ_2 ώστε $\Gamma \vdash P \triangleright \Delta_1$ και $\Gamma \vdash P \triangleright \Delta_2$.

(α′) Αν $P = \mathbf{0}$, τότε ο μόνος κανόνας του συστήματος ελέγχου τύπων που εφαρμόζεται είναι ο (Nil), που μας δίνει ότι $\Delta_1 = \Delta_2 = \emptyset$.

(β′) Αν $P = !P'$, τότε ο μόνος κανόνας του ελέγχου τύπων από τον οποίο μπορεί να έχουν προκύψει τα $\Gamma \vdash P \triangleright \Delta_1$ και $\Gamma \vdash P \triangleright \Delta_2$ είναι ο (Rep), που προϋποθέτει ότι $\Gamma \vdash P' \triangleright \Delta_1$ και $\Gamma \vdash P' \triangleright \Delta_2$. Από την επαγωγική υπόθεση, υπάρχει μοναδικό Δ' ώστε $\Gamma \vdash P' \triangleright \Delta'$, άρα $\Delta_1 = \Delta_2$.

(γ′) Αν $P = P' \mid P''$, τότε ο μόνος κανόνας του ελέγχου τύπων από τον οποίο μπορεί να έχουν προκύψει τα $\Gamma \vdash P \triangleright \Delta_1$ και $\Gamma \vdash P \triangleright \Delta_2$ είναι ο (ParP), που προϋποθέτει ότι $\Gamma \vdash P' \triangleright \Delta'_1$, $\Gamma \vdash P' \triangleright \Delta'_2$, $\Gamma \vdash P'' \triangleright \Delta''_1$ και $\Gamma \vdash P'' \triangleright \Delta''_2$, με $\Delta_1 = \Delta'_1 \uplus \Delta''_1$ και $\Delta_2 = \Delta'_2 \uplus \Delta''_2$. Από την επαγωγική υπόθεση, υπάρχουν μοναδικά $\Delta_{P'}$ και $\Delta_{P''}$ ώστε $\Gamma \vdash P' \triangleright \Delta_{P'}$ και $\Gamma \vdash P'' \triangleright \Delta_{P''}$, άρα $\Delta'_1 = \Delta'_2$ και $\Delta''_1 = \Delta''_2$, οπότε $\Delta_1 = \Delta_2$.

(δ′) Αν $P = (\nu\, x : T)P'$, τότε ο μόνος κανόνας του ελέγχου τύπων από τον οποίο μπορεί να έχουν προκύψει τα $\Gamma \vdash P \triangleright \Delta_1$ και $\Gamma \vdash P \triangleright \Delta_2$ είναι ο ResNP, που προϋποθέτει ότι $\Gamma \cdot x : T \vdash P' \triangleright \Delta_1$ και $\Gamma \cdot x : T \vdash P' \triangleright \Delta_2$. Από την επαγωγική υπόθεση, υπάρχει μοναδικό Δ' ώστε $\Gamma \vdash P' \triangleright \Delta'$, άρα $\Delta_1 = \Delta_2$.

(ε′) Αν $P = x\,\langle y \rangle\,.P'$, τότε ο μόνος κανόνας του ελέγχου τύπων από τον οποίο μπορεί να έχουν προκύψει τα $\Gamma \vdash P \triangleright \Delta_1$ και $\Gamma \vdash P \triangleright \Delta_2$ είναι ο (Out), που προϋποθέτει ότι $\Gamma \vdash x \triangleright G[T]$ (το $G[T]$ είναι μοναδικό, από την πρώτη περίπτωση της πρότασης), $\Gamma \cdot x : T \vdash P' \triangleright \Delta'_1$ και $\Gamma \cdot x : T \vdash P' \triangleright \Delta'_2$, με $\Delta_1 = \Delta'_1 \uplus \Delta_{\mathrm{w}}(G[T])$ και $\Delta_2 = \Delta'_2 \uplus \Delta_{\mathrm{w}}(G[T])$. Από την επαγωγική υπόθεση, υπάρχει μοναδικό Δ' ώστε $\Gamma \vdash P' \triangleright \Delta'$, άρα $\Delta'_1 = \Delta'_2$ και, από τη μοναδικότητα του $G[T]$, $\Delta_1 = \Delta_2$.

(στ′) Αν $P = [\![x\ op\ y]\!]P'$, τότε η απόδειξη είναι όμοια με της προηγούμενης περίπτωσης.

(ζ′) Αν $P = x(y : T).P'$, τότε η απόδειξη είναι όμοια με τις αποδείξεις των περιπτώσεων 2ε′ και 2δ′.

(η′) Αν $P = [x = y](P_1; P_2)$, τότε η απόδειξη είναι όμοια με τις αποδείξεις των περιπτώσεων 2ε′ και 2γ′.

3. Αποδεικνύεται τελείως ανάλογα με την περίπτωση των διεργασιών.

□

Με βάση την παραπάνω πρόταση, δικαιούμαστε να υλοποιήσουμε τις σχέσεις $\Gamma \vdash x \triangleright T$, $\Gamma \vdash P \triangleright \Delta$ και $\Gamma \vdash S \triangleright \Theta$ ως μερικές συναρτήσεις.

```
op _|-_ : GEnvironment Name ~> Type .
op _|-_ : GEnvironment Process ~> DEnvironment [frozen(2)] .
op _|-_ : GEnvironment System ~> ThInterface [frozen(2)] .

ceq GAMMA . X : T |- P = GAMMA |- P if not X in fn(P) .
ceq GAMMA . X : T |- S = GAMMA |- S if not X in fn(S) .

ceq GAMMA . X : T |- X = T
    if fg(T) subset groups(GAMMA) [label Name] .
```

```
eq GAMMA |- 0P = none-D [label NilP] .
ceq GAMMA . X : G[T] |- X (Y : T) . P =
    (GAMMA . X : G[T] . Y : T |- P) + Dr(T)
    if GAMMA . X : G[T] |- X == G[T] [label In] .
ceq GAMMA . X : G[T] |- X < Y >. P =
    (GAMMA . X : G[T] |- P) + Dw(G[T])
    if GAMMA . X : G[T] |- X == G[T] /\ GAMMA |- Y == T [label Out].
eq GAMMA |- (v X : T) P = GAMMA . X : T |- P [label ResNP] .
eq GAMMA |- ! P = GAMMA |- P [label Rep] .
ceq GAMMA |- [[X OP Y]] P = (GAMMA |- Y) OP Y + GAMMA |- P
    if GAMMA |- Y :: CtxtVar
    /\ GAMMA |- Y == GAMMA |- X [label CondA] .
eq GAMMA |- [X == Y] (P ; P') =
    (GAMMA |- [[X == Y]]P) + (GAMMA |- [[X =/= Y]]P') [label CondB].
eq GAMMA |- NEP1 | NEP2 =
    (GAMMA |- NEP1) + (GAMMA |- NEP2) [label ParP] .

eq GAMMA |- 0S = none-Th [label NilS] .
eq GAMMA |- (v G) P < U > =
    G [U] + (GAMMA . group: G |- P) [label ResGP] .
eq GAMMA |- (v R) S = R + (GAMMA . group: R |- S) [label ResGS] .
eq GAMMA |- (v X : T) S = GAMMA . X : T |- S [label ResNS] .
eq GAMMA |- NES1 | NES2 =
    (GAMMA |- NES1) ; (GAMMA |- NES2) [label ParS] .
```

Στις τρεις πρώτες γραμμές, ορίζουμε τους τελεστές· παρατηρούμε το σύμβολο ~>, που δηλώνει πως η συνάρτηση είναι μερική και την ιδιότητα **frozen** που απαγορεύει στα συστήματα και τις διεργασίες να κάνουν μεταβάσεις κατά τη διάρκεια του ελέγχου τύπων. Οι γραμμές 5 και 6 δικαιολογούνται από το λήμμα 3.32 και χρησιμεύουν σε περιπτώσεις που ένα όνομα χρησιμοποιείται για να συμβολίσει περισσότερα από ένα κανάλια σε μια διεργασία ή ένα σύστημα, όπως για παράδειγμα το x στην $x(y : T).(\nu\, x : T)\overline{y}\,\langle x\rangle\,.\mathbf{0}$. Ακολουθούν οι κανόνες του ελέγχου τύπων, οι οποίοι αποτελούν μεταγραφές των κανόνων του ορισμού 3.15· στις γραμμές 8 και 9 έχουμε τον κανόνα που αφορά ονόματα, στις γραμμές 11 έως 26 τους κανόνες που αφορούν διεργασίες και στις γραμμές 28 έως 34 τους κανόνες που αφορούν συστήματα. Οι κανόνες In και Out απαιτούν το κανάλι μέσω του οποίου γίνεται η επικοινωνία να έχει τύπο $G[T]$, χωρίς όμως να προκύπτει κάποιος περιορισμός για το G από τη δομή της διεργασίας· για να ξεπεράσουμε την εν λόγω δυσκολία, εκμεταλλευόμαστε το γεγονός πως, λόγω και του κανόνα Name, για να περνάει το $x(y : T).P'$ (ή το $\overline{x}\,\langle y\rangle\,.P'$) τον έλεγχο τύπων, πρέπει το περιβάλλον Γ να περιέχει μια (μοναδική) καταχώρηση $x : G[T]$· αυτό βέβαια δεν αναιρεί την ανάγκη να εξετάσουμε αν το $G[T]$ προκύπτει όντως ως τύπος για το x με βάση το περιβάλλον Γ (αν δηλαδή οι ελεύθερες ομάδες του τύπου καταγράφονται στο περιβάλλον Γ).

Μένει τώρα να υλοποιήσουμε τα σχετικά με το πότε μια διεπαφή Θ σέβεται μια πολιτική. Για τη σχέση $\mathcal{P} \vDash \Theta$, κάνουμε επαγωγή στη δομή του Θ. Αν το Θ είναι η κενή διεπαφή, τότε σέβεται τετριμμένα την πολιτική. Αν το Θ περιέχει μόνο μία καταχώρηση της μορφής $t \gg \langle H^{\downarrow}, \tilde{p}\rangle$, τότε υπάρχουν

δύο περιπτώσεις: είτε η πολιτική έχει μόνο μία καταχώρηση, οπότε πρέπει η καταχώρηση να είναι της μορφής $t \gg H, \pi$ και το αποτέλεσμα της συνάρτησης perm να είναι λιγότερο αυστηρό από το $\tilde{p}$, είτε η πολιτική έχει πολλές καταχωρήσεις, οπότε πρέπει το προηγούμενο να ισχύει για μία από αυτές. Τέλος, αν το Θ έχει πολλές καταχωρήσεις, πρέπει η καθεμιά από αυτές να ικανοποιεί την πολιτική.

```
1  op _|=_ : Policy ThInterface -> Bool [prec 50] .
2  eq POL |= none-Th = true .
3  eq BT >> H, PF |= BT >> < HTH, PS > =
4      PS <= perms(PF, H, groups(HTH), purpose(HTH)) .
5  eq BT >> H, PF ; POL |= BT >> < HTH, PS > =
6      BT >> H, PF |= BT >> < HTH, PS > .
7  eq POL |= NETH1 ; NETH2 = POL |= NETH1 and POL |= NETH2 .
8  eq POL |= TH = false [owise] .
```

Όσον αφορά τη συνάρτηση $\text{perms}_{\pi}(H, \tilde{G}, u)$, ο ορισμός της ακολουθεί το μαθηματικό ορισμό, με τη διαφορά ότι χρησιμοποιούμε τη βοηθητική συνάρτηση $perms για να υλοποιήσουμε το $\bigcup_{H_i \in \tilde{H}}$. Επιπλέον, χρησιμοποιούμε τον τελεστή σφάλματος (έχει είδος αλλά όχι Τύπο) incompatibleHierarchy για την περίπτωση που η perms επιστρέφει $\perp$.

```
op perms : PermissionFunction Hierarchy Set{Group} Purpose
    ~> Set{Permission} .
op $perms : PermissionFunction Set{Hierarchy} Set{Group} Purpose
    -> Set{Permission} .
op incompatibleHierarchy : Hierarchy Set{Group}
    -> [Set{Permission}].
eq perms(PF, nil, GS, U) = empty .
eq perms(PF, H, empty, U) = empty .
ceq perms(PF, G : US [HS], GS, U) =
    incompatibleHierarchy(G : US[HS], GS) if not G in GS .
eq perms(PF, G : US [HS], GS, U) =
    union(if U in US then PF(U, G) else empty fi,
        $perms(PF, if U in US then U -> HS else HS fi,
            intersection(groups(HS), GS), U)) .
eq $perms(PF, empty, GS, U) = empty .
eq $perms(PF, (H, HS), GS, U) = union(
    if root(H) subset GS then perms(PF, H, GS, U) else empty fi,
    $perms(PF, HS, GS, U)) .
```

4.4 Παραδείγματα χρήσης της υλοποίησης

Έχοντας υλοποιήσει όλα όσα απαιτούνται για τον έλεγχο τύπων, μένει να δούμε πώς μπορούμε να κάνουμε υποστασιοποιήσεις σε δεδομένες εφαρμογές και να τις χρησιμοποιούμε για να αποδεικνύουμε την ασφάλεια συστημάτων του π-λογισμού.

Θα χρησιμοποιήσουμε την πολιτική ιδιωτικότητας του παραδείγματος 3.2. Αρχικά, δίνουμε τους ρόλους, τους χρήστες, τους σκοπούς και τους βασικούς τύπους.

```
ops Comp&Clients ThirdParty Clients Company AdminDpt OrderDpt
    MarketingDpt PurchaseDpt ShippingDpt ThirdParty : -> Role .
ops Alice Bob : -> User .
op B.Address : -> BasicType .
ops purchase analysis marketing : -> Purpose .
```

Έπειτα, για κάθε μεταβλητή πλαισίου δημιουργούμε μία σταθερά τύπου CtxtVar που θα αντιστοιχεί στη μεταβλητή, καθώς και έναν υποτύπο του CtxtVarValue στο οποίο θα συμπεριλάβουμε τις τιμές της. Η μεταβλητή πλαισίου αντιστοιχίζεται στο σύνολο των τιμών της μέσω του τελεστή domain .

```
sort B.ConsentValue .
subsort B.ConsentValue < CtxtVarValue .
op B.Consent : -> CtxtVar .
ops Yes No : -> B.ConsentValue [ctor] .
eq domain(B.Consent) = Yes No .

sort B.AgeValues .
subsort B.AgeValues < CtxtVarValue .
op B.Age : -> CtxtVar .
ops 0-17 18-30 31-60 over60 : -> B.AgeValues .
eq domain(B.Age) = 0-17 18-30 31-60 over60 .
```

Όσον αφορά τις συναρτήσεις απόδοσης δικαιωμάτων, ορίζονται όπως φαίνεται πιο κάτω. Να σημειωθεί εδώ πως –τουλάχιστον όπως έχουν οριστεί– οι επιμέρους αποδόσεις δικαιωμάτων δεν μπορούν να έχουν μεταβλητές στη θέση του σκοπού ή της ομάδας, άρα δεν υποστηρίζεται άμεσα απόδοση δικαιώματος για οποιονδήποτε σκοπό (όπως πχ είχαμε δώσει στον Bob στο παράδειγμα 3.2)· αντ' αυτού, οφείλουμε να συμπεριλαμβάνουμε μία ξεχωριστή καταχώρηση για κάθε δυνατό σκοπό (εδώ τυχαίνει τον Bob να αφορά μόνο ένας σκοπός) ή να ορίσουμε ιεραρχία ανάμεσα στους σκοπούς που να περιλαμβάνει ως αντικείμενο έναν οικουμενικό σκοπό.

```
ops pmax pAlice : -> Set{Permission} .
eq pmax = read write access disc Comp&Clients disc ThirdParty
    disc Clients disc Company disc AdminDpt disc OrderDpt
    disc MarketingDpt disc PurchaseDpt disc ShippingDpt
    disc ThirdParty disc Alice disc Bob .
eq pAlice = read access disc Comp&Clients disc Company
    disc OrderDpt disc MarketingDpt disc PurchaseDpt
    disc AdminDpt disc ShippingDpt disc Alice disc Bob .

ops p`{B.Age`} p`{B.Address`} p`{B.Consent`} : ->
    PermissionFunction .

eq p{B.Age} =
    p(analysis, AdminDpt) = access read,
    p(purchase, PurchaseDpt) = access read,
    p(marketing, MarketingDpt) = access read,
    p(purchase, Bob) = pmax .
```

```
eq p{B.Consent} =
    p(analysis, AdminDpt) = access read,
    p(marketing, MarketingDpt) = access read,
    p(purchase, Bob) = pmax .

eq p{B.Address} =
    p(purchase, PurchaseDpt) = access if B.Age =/= 0-17
                               disc OrderDpt if B.Age =/= 0-17,
    p(purchase, ShippingDpt) = access read,
    p(marketing, MarketingDpt) = access if B.Age =/= 0-17
           disc ThirdParty if B.Age =/= 0-17 /\ B.Consent == Yes,
    p(purchase, Alice) = pAlice,
    p(purchase, Bob) = pmax .
```

Για την ιεραρχία και την πολιτική ιδιωτικότητας, η σύνταξη δεν διαφέρει ιδιαίτερα από τη μαθηματική περιγραφή.

```
op H : -> Hierarchy .
eq H =
    Comp&Clients[
        Clients : purchase [
            Alice [], Bob[]
        ],
        Company [
            AdminDpt : analysis,
            OrderDpt : purchase[
                PurchaseDpt [],
                ShippingDpt []
            ],
            MarketingDpt : marketing
        ]
     ].

op policy : -> Policy .
eq policy = B.Address >> H, p{B.Address} ;
            B.Age >> H, p{B.Age} ;
            B.Consent >> H, p{B.Consent} .
```

Έχοντας καταγράψει την πολιτική ιδιωτικότητας, το επόμενο βήμα είναι να ορίσουμε κάποια συστήματα (ή διεργασίες) που θα ελεγχθούν όσον αφορά το αν σέβονται την πολιτική. Θα χρησιμοποιήσουμε τα S_1 και S_2 από το παράδειγμα 3.3· και εδώ τα πράγματα είναι αρκετά όμοια με τη μαθηματική περιγραφή.

```
ops S1 S2 : -> System .

ops sendaddr address order getage bobage addr a : -> Name .
op T1 : -> Type .
eq T1 = Comp&Clients[B.Address] .

eq S1 = (v Comp&Clients)(v sendaddr : Comp&Clients[T1])(
          (v Clients)(v Alice)sendaddr < address >. 0P < purchase >
```

```
|( v Company)(v OrderDpt)(v order : OrderDpt[T1])
 (v getage : Company[B.Age])(
       (v PurchaseDpt)![bobage =/= 0-17]sendaddr(addr : T1).
                       order < addr >. 0P < purchase >
    | (v ShippingDpt)! order(addr : T1).
                       addr(a : B.Address). 0P < purchase >
 )
) .
```

ops sendtotp age readc cons addr : −> Name .
op T2 : −> Type .
eq T2 = ThirdParty[ThirdParty[B.Address]] .

```
eq S2 = (v Comp&Clients)(v ThirdParty)(v sendtotp : T2)(v Company)
          (v MarketingDpt)[[age =/= 0-17]]readc(cons : B.Consent).
          (v addr : ThirdParty[B.Address])sendtotp < addr >. 0P
       < marketing > .
```

Το μόνο που μένει για τον έλεγχο τύπων είναι να καθορίσουμε τα περιβάλλοντα Γ. Σύμφωνα με τα λήμματα 3.28 και 3.32, ένα περιβάλλον Γ αρκεί να περιλαμβάνει μία καταχώρηση για κάθε ελεύθερο όνομα και για κάθε ελεύθερη ομάδα του προς εξέταση συστήματος. Έχουμε ήδη δει ποια περιβάλλοντα χρειαζόμαστε στο παράδειγμα 3.4.

```
ops Gamma1 Gamma2 : −> GEnvironment .
eq Gamma1 = address : Comp&Clients[B.Address] .
            bobage : B.Age . 0-17 : B.Age .
eq Gamma2 = age : B.Age . 0-17 : B.Age .
            readc : Comp&Clients[B.Consent] .
```

Μπορούμε τώρα να γράψουμε στο prompt

```
> red policy |= Gamma1 |- S1 .
```

το οποίο δίνει την απάντηση

```
rewrites: 4038 in 4ms cpu (3ms real) (933000 rewrites/second)
result Bool: true
```

από την οποία συμπεραίνουμε ότι το S_1 σέβεται την πολιτική ιδιωτικότητας. Κάνουμε το ίδιο και για το S_2

```
> red policy |= Gamma2 |- S2 .
```

από όπου παίρνουμε

```
rewrites: 2027 in 4ms cpu (2ms real) (506750 rewrites/second)
result Bool: false
```

δηλαδή για το δεδομένο περιβάλλον Γ ο έλεγχος τύπων αποτυγχάνει. Με ένα επιχείρημα όπως του παραδείγματος 3.4 μπορούμε –εν προκειμένω– να διαπιστώσουμε ότι αυτό συνέβη επειδή το S_2 όντως δεν σέβεται την πολιτική ιδιωτικότητας.

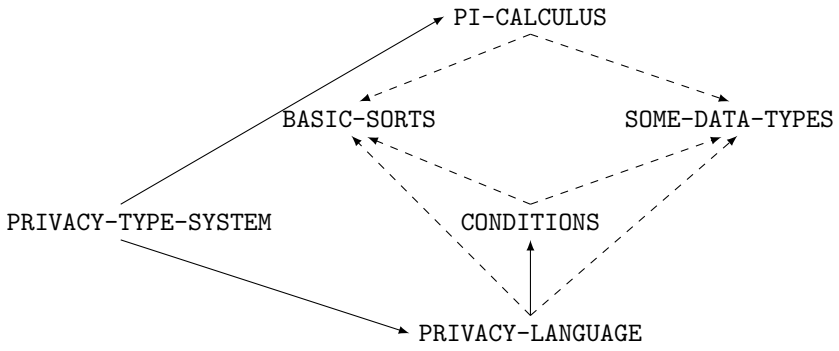
Παράρτημα Α΄

Ο πλήρης κώδικας της υλοποίησης

Εδώ παρουσιάζεται αναλυτικά ο κώδικας που περιγράφεται στις ενότητες 4.1, 4.2 και 4.3, χωρίς τις απλοποιητικές παραδοχές που χρησιμοποιήθηκαν εκεί.

Συγκεκριμένα, ο κώδικας οργανώνεται σε αρθρώματα όπως φαίνεται στο σχήμα Α΄.1. Τα `PI-CALCULUS` και `PRIVACY-TYPE-SYSTEM` περιέχουν τον κώδικα που περιγράφηκε στις ενότητες 4.2 και 4.3 αντίστοιχα, ενώ τα `CONDITIONS` και `PRIVACY-LANGUAGE` περιέχουν τον κώδικα που περιγράφηκε στην ενότητα 4.1. Δεδομένου ότι χρησιμοποιήσαμε μόνο κανόνες της εξισωτικής λογικής, όλα τα αρθρώματα είναι συναρτησιακά. Για αισθητικούς λόγους, επιλέχθηκε όλα τα σύνολα να οριστούν στο `SOME-DATA-TYPES`· αυτό απαιτούσε τη δημιουργία του `BASIC-SORTS`, το οποίο περιλαμβάνει τις δηλώσεις των Τύπων που θα χρησιμοποιηθούν σε σύνολα. Επιπλέον, οι κοινοί Τύποι των `PRIVACY-LANGUAGE` και `PI-CALCULUS` (για παράδειγμα το `Group`) έπρεπε να δηλωθούν σε κάποιο άρθρωμα που θα συμπεριλαμβανόταν και στα δύο, διότι διαφορετικά το `PRIVACY-TYPE-SYSTEM` θα τα αντιμετώπιζε ως δύο διαφορετικούς αλλά ομώνυμους Τύπους (δηλαδή δύο διαφορετικές «εκδόσεις» του Τύπου `Group`)· οι εν λόγω Τύποι συμπεριλαμβάνονται επίσης στο `BASIC-SORTS`.

Τέλος, για να χρησιμοποιήσουμε τον ενσωματωμένο στην Maude κώδικα για σύνολα και λόγω του τρόπου που υλοποιούνται οι παραμετρικοί τύποι δεδομένων, είναι απαραίτητο να ορίσουμε ένα view για κάθε Τύπο που θα χρησιμοποιήσουμε σε σύνολα (περισσότερες πληροφορίες στην ενότητα 6.3 του [7]). Η μετονομασία των τελεστών σε κάποια σύνολα που περιγράφηκε στην αρχή του κεφαλαίου 4 υλοποιείται με τον τελεστή *.



Σχήμα Α΄.1: Οι ρητά καθορισμένες σχέσεις συμπερίληψης ανάμεσα στα επιμέρους αρθρώματα. Το άρθρωμα στην αρχή του βέλους περιλαμβάνει εκείνο στο πέρας του βέλους. Τα διακεκομμένα βέλη δηλώνουν τη σχέση `extending`, ενώ τα συνεχή τη σχέση **protecting**.

```
fmod BASIC-SORTS is
    sorts BasicType User Role Group Purpose Permission Hierarchy
        UnconditionalPermission ConditionalPermission Condition
        CtxtVar CtxtVarValue CtxtVarOperator Type Name .
    subsorts User Role < Group .
    subsorts ConditionalPermission UnconditionalPermission
        < Permission .
    subsorts CtxtVar < BasicType < Type .
    subsort CtxtVarValue < Name .

    ops == =/= : -> CtxtVarOperator [ctor] .
endfm

view Type from TRIV to BASIC-SORTS is
    sort Elt to Type .
endv

view Group from TRIV to BASIC-SORTS is
    sort Elt to Group .
endv

view Purpose from TRIV to BASIC-SORTS is
    sort Elt to Purpose .
endv

view Permission from TRIV to BASIC-SORTS is
    sort Elt to Permission .
endv

view Hierarchy from TRIV to BASIC-SORTS is
    sort Elt to Hierarchy .
endv
```

```
view Name from TRIV to BASIC-SORTS is
    sort Elt to Name .
endv

view CtxtVar from TRIV to BASIC-SORTS is
    sort Elt to CtxtVar .
endv

view CtxtVarValue from TRIV to BASIC-SORTS is
    sort Elt to CtxtVarValue .
endv

view Condition from TRIV to BASIC-SORTS is
    sort Elt to Condition .
endv

fmod SOME-DATA-TYPES is
    protecting SET{Condition} .
    protecting SET{Hierarchy} .
    protecting SET{Group} .
    protecting SET{Purpose} .
    protecting SET{Permission} * (
        op _,_ to __ [prec 28]
    ) .
    protecting SET{Type} .
    protecting SET{Name} .
    protecting SET{CtxtVar} * (
        op _,_          to __ ,
        op empty        to empty-X ,
        op insert       to insert-X ,
        op delete       to delete-X ,
        op _in_         to _X-in_ ,
        op |_|          to card-X ,
        op $card        to $card-X ,
        op union        to union-X ,
        op intersection to intersection-X ,
        op $intersect   to $intersect-X ,
        op _\_          to _diff-X_ ,
        op $diff        to $diff-X ,
        op _subset_     to _subset-X_ ,
        op _psubset_    to _psubset-X_
    ) .
    protecting SET{CtxtVarValue} * (
        op _,_          to __ ,
        op empty        to empty-V ,
        op insert       to insert-V ,
        op delete       to delete-V ,
        op _in_         to _V-in_ ,
        op |_|          to card-V ,
```

```
        op $card       to $card-V ,
        op union       to union-V ,
        op intersection to intersection-V ,
        op $intersect  to $intersect-V ,
        op _\_         to _diff-V_ ,
        op $diff       to $diff-V ,
        op _subset_    to _subset-V_ ,
        op _psubset_   to _psubset-V_
    ) .
endfm

fmod CONDITIONS is
    extending BASIC-SORTS .
    extending SOME-DATA-TYPES .

    vars X Y : CtxtVar .
    vars XS YS : Set{CtxtVar} .
    var OP : CtxtVarOperator .
    vars VAL VAL1 VAL2 VAL' : CtxtVarValue .
    vars COND COND' COND1 COND2 : Condition .
    var CONDS : Set{Condition} .

    op domain : CtxtVar -> Set{CtxtVarValue} .

    op ___ : CtxtVar CtxtVarOperator CtxtVarValue
        -> [Condition] [ctor prec 14].
    cmb X OP VAL : Condition if VAL V-in domain(X) .

    op _/\_ : Condition Condition -> Condition
        [ctor assoc comm prec 15].

    op vars : Condition -> Set{CtxtVar} .
    eq vars(X OP VAL) = X .
    eq vars(COND1 /\ COND2) = union-X(vars(COND1), vars(COND2)) .

    op _<=_ : Condition Condition -> Bool .
    eq COND <= COND = true .
    eq COND <= COND' = vars(COND') subset-X vars(COND)
        and $stricter-than(COND, COND', vars(COND')) .
    op $stricter-than : Condition Condition Set{CtxtVar} -> Bool .
    eq $stricter-than(COND, COND', empty-X) = true .
    eq $stricter-than(COND, COND', (X XS)) =
        allowed(COND, X) subset-V allowed(COND', X)
        and $stricter-than(COND, COND', XS) .

    op allowed : Condition CtxtVar -> Set{CtxtVarValue} .
    eq allowed(X == VAL, Y) = if X == Y then VAL else domain(Y) fi .
    eq allowed(X =/= VAL, Y) =
        if X == Y then delete-V(VAL, domain(X)) else domain(Y) fi .
    eq allowed(COND1 /\ COND2, X) =
```

```
        intersection-V(allowed(COND1, X), allowed(COND2, X)) .
endfm

fmod PRIVACY-LANGUAGE is
    extending BASIC-SORTS .
    extending SOME-DATA-TYPES .
    protecting CONDITIONS .

    sorts PermissionFunction Policy .

    var BT : BasicType .
    var G : Group .
    var R : Role .
    var USER : User .
    vars Gs Gs' : Set{Group} .
    var U : Purpose .
    var US : Set{Purpose} .
    vars P P1 P2 P' : Permission .
    vars PS PS' PS1 PS2 : Set{Permission} .
    vars H H1 H2 : Hierarchy .
    vars HS HS' : Set{Hierarchy} .
    vars POL POL1 POL2 : Policy .
    vars PF PF1 PF2 : PermissionFunction .
    vars COND COND' : Condition .
    var CONDS : Set{Condition} .

    ---------- Define Permissions ----------
    ops read write access : -> UnconditionalPermission [ctor].
    op disc_ : Group -> UnconditionalPermission [ctor prec 15].
    op _if_ : UnconditionalPermission Condition
        -> ConditionalPermission [ctor prec 16] .

    op _+_ : Condition Set{Permission} -> Set{Permission} [prec 17] .
    eq COND + P+:UnconditionalPermission =
        P+:UnconditionalPermission if COND .
    eq COND + P if COND' = P if COND /\ COND' .
    eq COND + empty = empty .
    eq COND + (P PS) = (COND + P) (COND + PS) .

    op _<=_ : Set{Permission} Set{Permission} -> Bool .
    eq empty <= PS = true .
    eq P if COND PS <= P if COND' PS' =
        COND <= COND' and PS <= P if COND' PS' .
    eq P if COND PS <= P PS' = PS <= P PS' .
    eq P PS <= P PS' = PS <= P PS' .
    eq PS1 <= PS2 = false [owise] .

    ---------- Define Permission Function ----------
    op (p'(_',_')=_) : Purpose Group Set{Permission}
        -> PermissionFunction [ctor prec 29] .
```

```
op _‘,_ : PermissionFunction PermissionFunction
    -> PermissionFunction [ctor assoc comm prec 30] .

op (_‘(_‘,_‘)) : PermissionFunction Purpose Group
    -> Set{Permission} .
eq (p(U, G) = PS)(U, G) = PS .
eq (PF1, PF2)(U,G) = union(PF1(U,G), PF2(U,G)) .
eq PF(U,G) = empty [owise] .

---------- Define Hierarchy ----------
op nil : -> Hierarchy [ctor].
op _:_‘[_‘] : Group Set{Purpose} NeSet{Hierarchy} ~> Hierarchy
    [ctor ].

op _:_ : Group Set{Purpose} -> Hierarchy .
eq G : US = G : US[nil] .
op _‘[_‘] : Group NeSet{Hierarchy} ~> Hierarchy .
eq G[HS] = G : empty[HS] .
op _‘[‘] : Group -> Hierarchy .
eq G []  = G : empty[nil] .

op cyclicalHierarchyError : -> [Hierarchy] .

cmb R : US[HS] : Hierarchy if not R in groups(HS) .
cmb USER : US[HS] : Hierarchy if HS == nil .
ceq G : US[HS] = cyclicalHierarchyError if G in groups(HS) .

op groups : Set{Hierarchy} -> Set{Group} .
eq groups(nil) = empty .
eq groups(G : US[HS]) = insert(G, groups(HS)).
eq groups(empty) = empty .
eq groups((H, HS)) = union(groups(H), groups(HS)) .

op root : Hierarchy -> Set{Group} .
eq root(nil) = empty .
eq root(G : US[HS]) = G .

op _->_ : Purpose Set{Hierarchy} -> Set{Hierarchy} .
eq U -> nil = nil .
eq U -> G : US[HS] = G : insert(U, US)[HS] .
eq U -> (H, HS) = (U -> H, U -> HS) .

---------- Define Policy ----------
op _>>_‘,_ : BasicType Hierarchy PermissionFunction
    -> Policy [ctor prec 35] .
op _;_ : Policy Policy ~> Policy [ctor assoc comm prec 40] .

op multiplePoliciesForType : Type -> [Policy] .

op types : Policy -> Set{Type} .
```

```
    eq types(BT >> H , PF) = BT .
    eq types(POL1 ; POL2) = union(types(POL1), types(POL2)) .

    cmb POL1 ; POL2 : Policy
        if intersection(types(POL1), types(POL2)) = empty .
    eq BT >> H1, PF1 ; BT >> H2, PF2 = multiplePoliciesForType(BT) .
endfm

fmod PI-CALCULUS is
    extending BASIC-SORTS .
    extending SOME-DATA-TYPES .

    sorts Process NeProcess System NeSystem .
    subsort NeProcess < Process .
    subsort NeSystem < System .

    vars X Y : Name .
    var T : Type .
    vars P P1 P2 : Process .
    vars NEP1 NEP2 : NeProcess .
    var S : System .
    vars NES1 NES2 : NeSystem .
    var OP : CtxtVarOperator .
    var G : Group .
    var R : Role .
    var U : Purpose .

    ---------------- Types ----------------
    op _`[_`] : Group Type -> Type [prec 15] .

    op fg : Type -> Set{Group} .
    eq fg(BT:BasicType) = empty .
    eq fg(G[T]) = insert(G, fg(T)) .

    ---------------- Processes ----------------
    op 0P : -> Process [ctor] .

    op !_ : Process -> NeProcess [frozen ctor prec 17] .

    op _|_ : Process Process -> Process
        [frozen ctor assoc comm id: 0P prec 19] .
    op _|_ : Process NeProcess -> NeProcess [ditto] .

    op (`(v_:_`)_) : Name Type Process ~> NeProcess
        [frozen(3) ctor prec 17] .
    cmb (v X : T) P : NeProcess if not X :: CtxtVarValue .

    op (_`(_:_`)._) : Name Name Type Process ~> NeProcess
        [frozen(4) ctor prec 17] .
    cmb X(Y : T). P : NeProcess if not Y :: CtxtVarValue .
```

```
op _<_>._ : Name Name Process -> NeProcess [frozen(3) ctor prec 17].

op (`[_==_`]`(_;_`)) : Name CtxtVarValue Process Process
    -> NeProcess [frozen(3 4) ctor] .

op `[`[___`]`]_ : Name CtxtVarOperator CtxtVarValue Process
    -> NeProcess [frozen(4) ctor prec 17] .

op `[___`]_ : Name CtxtVarOperator CtxtVarValue Process
    -> NeProcess [frozen(4) prec 17] .
eq [X == Y]P = [X == Y] (P ; 0P) .
eq [X =/= Y]P = [X == Y] (0P ; P) .

ops fn bn : Process -> Set{Name} .

eq fn(0P) = empty .
eq fn(! P) = fn(P) .
eq fn(NEP1 | NEP2) = union(fn(NEP1), fn(NEP2)) .
eq fn((v X : T) P) = delete(X, fn(P)) .
eq fn(X (Y : T). P) = insert(X, delete(Y, fn(P))) .
eq fn(X < Y >. P) = insert(X, insert(Y, fn(P))) .
eq fn([X == Y] (P1 ; P2)) =
    insert(X, insert(Y, union(fn(P1), fn(P2)))) .
eq fn([[X OP Y]] P) = insert(X, insert(Y, fn(P))) .

eq bn(0P) = empty .
eq bn(! P) = bn(P) .
eq bn(NEP1 | NEP2) = union(bn(NEP1), bn(NEP2)) .
eq bn((v X : T) P) = insert(X, bn(P)) .
eq bn(X (Y : T). P) = insert(Y, bn(P)) .
eq bn(X < Y >. P) = bn(P) .
eq bn([X == Y] (P1 ; P2)) = union(bn(P1), bn(P2)) .
eq bn([[X OP Y]] P) = bn(P) .

op fg : Process -> Set{Group} .
eq fg(0P) = empty .
eq fg(! P) = fg(P) .
eq fg(NEP1 | NEP2) = union(fg(NEP1), fg(NEP2)) .
eq fg((v X : T) P) = union(fg(P), fg(T)) .
eq fg(X (Y : T). P) = union(fg(P), fg(T)) .
eq fg(X < Y >. P) = fg(P) .
eq fg([X == Y] (P1 ; P2)) = union(fg(P1), fg(P2)) .
eq fg([[X OP Y]] P) = fg(P) .

---------- Systems ----------
op 0S : -> System [ctor] .

op _|_ : System System -> System
    [frozen ctor assoc comm id: 0S prec 25] .
```

```
    op _|_ : System NeSystem -> NeSystem [ditto] .

    op (`(v_:_`)_) : Name Type System ~> NeSystem
        [frozen(3) ctor prec 23].
    cmb (v X : T) S : NeSystem if not X :: CtxtVarValue .

    op (`(v_`)_<_>) : Group Process Purpose -> NeSystem
        [frozen(2) ctor prec 21] .

    op (`(v_`)_) : Role System -> NeSystem [frozen(2) ctor prec 23] .

    ops fn bn : System -> Set{Name} .

    eq fn(0S) = empty .
    eq fn(NES1 | NES2) = union(fn(NES1), fn(NES2)) .
    eq fn((v X : T) S) = delete(X, fn(S)) .
    eq fn((v G) P < U >) = fn(P) .
    eq fn((v R) S) = fn(S) .

    eq bn(0S) = empty .
    eq bn(NES1 | NES2) = union(bn(NES1), bn(NES2)) .
    eq bn((v X : T) S) = insert(X, bn(S)) .
    eq bn((v G) P < U >) = bn(P) .
    eq bn((v R) S) = bn(S) .

    ops fg bg : System -> Set{Group} .

    eq fg(0S) = empty .
    eq fg(NES1 | NES2) = union(fg(NES1), fg(NES2)) .
    eq fg((v X : T) S) = union(fg(S), fg(T)) .
    eq fg((v G) P < U >) = delete(G, fg(P)) .
    eq fg((v R) S) = delete(R, bg(S)) .

    eq bg(0S) = empty .
    eq bg(NES1 | NES2) = union(bg(NES1), bg(NES2)) .
    eq bg((v X : T) S) = bg(S) .
    eq bg((v G) P < U >) = G .
    eq bg((v R) S) = insert(R, bg(S)) .
endfm

fmod PRIVACY-TYPE-SYSTEM is
    protecting PRIVACY-LANGUAGE .
    protecting PI-CALCULUS .

    sorts GEnvironment DEnvironment NeDEnvironment
        InterfaceHierarchy ThInterface NeThInterface .
    subsort NeDEnvironment < DEnvironment .
    subsort NeThInterface < ThInterface .
    subsort InterfaceHierarchy < Hierarchy .
```

```
vars X Y : Name .
vars P P' P1 P2 : Process .
vars NEP1 NEP2 : NeProcess .
vars S S1 S2 : System .
vars NES1 NES2 : NeSystem .

var OP : CtxtVarOperator .
var BT : BasicType .
vars T T' : Type .
var TS : Set{Type} .
var R : Role .
vars G G' : Group .
var GS : Set{Group} .
var U : Purpose .
var US : Set{Purpose} .
vars PS PS1 PS2 : Set{Permission} .
vars COND COND' : Condition .
var CONDS : Set{Condition} .
var H : Hierarchy .
vars HS HS' : Set{Hierarchy} .
var POL : Policy .
var PF : PermissionFunction .

vars D D1 D2 : DEnvironment .
vars NED NED1 NED2 : NeDEnvironment .
vars GAMMA GAMMA1 GAMMA2 : GEnvironment .
vars TH TH1 TH2 : ThInterface .
vars NETH1 NETH2 : NeThInterface .
var HTH : InterfaceHierarchy .

---------- Define GEnvironment ----------
op none-G : -> GEnvironment [ctor] .
op _:_ : Name Type -> GEnvironment [ctor prec 35] .
op group:_ : Group -> GEnvironment [ctor prec 35] .
op _._ : GEnvironment GEnvironment -> GEnvironment
    [ctor assoc comm id: none-G prec 40] .

op nameAppearsTwice : Name -> [GEnvironment] .
eq X : T . X : T' = nameAppearsTwice(X) .
op groupAppearsTwice : Group -> [GEnvironment] .
eq group: G . group: G = groupAppearsTwice(G) .

op groups : GEnvironment -> Set{Group} .
eq groups(none-G) = empty .
eq groups(X : T) = empty .
eq groups(group: G) = G .
eq groups(GAMMA1 . GAMMA2) = union(groups(GAMMA1), groups(GAMMA2)).

---------- Define DEnvironment ----------
op none-D : -> DEnvironment [ctor] .
```

```
op _:_ : BasicType Set{Permission} -> NeDEnvironment
    [ctor prec 35] .
op _._ : DEnvironment DEnvironment ~> DEnvironment
    [ctor assoc comm id: none-D prec 40] .
op _._ : NeDEnvironment DEnvironment ~> NeDEnvironment
    [ctor assoc comm id: none-D prec 40] .
cmb NED1 . NED2 : NeDEnvironment
    if intersection(types(NED1), types(NED2)) = empty .

op types : DEnvironment -> Set{Type} .
eq types(none-D) = empty .
eq types(BT : PS) = BT .
eq types(D1 . D2) = union(types(D1), types(D2)) .

op permissions : DEnvironment BasicType -> Set{Permission} .
eq permissions(BT : PS . D, BT) = PS .
eq permissions(D, BT) = empty [owise] .

---------- Define InterfaceHierarchy ----------
op _`[_`] : Group Purpose -> InterfaceHierarchy [ctor] .
op _`[_`] : Group InterfaceHierarchy -> InterfaceHierarchy
    [ctor].
eq G[US] = G : US [nil] .
cmb G : US [HS] : InterfaceHierarchy
    if (US == empty and |HS| == 1 and HS =/= nil)
    or (HS == nil and |US| == 1) .

op purpose : InterfaceHierarchy -> Purpose .
eq purpose(G : U [nil]) = U .
eq purpose(G : empty [H]) = purpose(H) .

---------- Define ThInterface ----------
op none-Th : -> ThInterface [ctor] .
op _>>`<_`,_> : BasicType InterfaceHierarchy Set{Permission}
    -> NeThInterface [ctor prec 35] .
op _;_ : ThInterface ThInterface -> ThInterface
    [ctor assoc comm id: none-Th prec 40] .
op _;_ : NeThInterface ThInterface -> NeThInterface [ditto] .

---------- Auxuliary functions ----------
op Dr : Type -> DEnvironment .
eq Dr(BT) = BT : read .
eq Dr(G [BT]) = BT : access .
eq Dr(T) = none-D [owise] .

op Dw : Type -> DEnvironment .
eq Dw(G[BT]) = BT : write .
eq Dw(G [G' [BT]]) = BT : disc G .
eq Dw(T) = none-D [owise] .
```

```
op _+_ : Condition DEnvironment -> DEnvironment .
eq COND + none-D = none-D .
eq COND + BT : PS = BT : COND + PS .
eq COND + (D1 . D2) = (COND + D1) . (COND + D2) .

op _+_ : DEnvironment DEnvironment -> DEnvironment
    [assoc comm id: none-D prec 41] .
op _+_ : NeDEnvironment DEnvironment -> NeDEnvironment [ditto].
eq NED1 + NED2 =
    $+(NED1, NED2, union(types(NED1), types(NED2))) .
op $+ : NeDEnvironment NeDEnvironment Set{Type}
    -> NeDEnvironment .
eq $+(D1, D2, empty) = none-D .
eq $+(D1, D2, (BT, TS)) = $+(D1, D2, TS) .
    BT : union(permissions(D1, BT), permissions(D2,BT)) .

op _`[_`]+_ : Group Purpose DEnvironment -> ThInterface [prec 35] .
op _`[_`]+_ : Group Purpose NeDEnvironment
    -> NeThInterface [ditto].
eq G [ U ] + none-D = none-Th .
eq G [ U ] + (BT : PS) = BT >> < G [U] , PS > .
eq G [ U ] + (D1 . D2) = G [ U ] + D1 ; G [ U ] + D2 .

op _+_ : Group ThInterface -> ThInterface [prec 35] .
op _+_ : Group NeThInterface -> NeThInterface [ditto] .
eq G + none-Th = none-Th .
eq G + (BT >> < HTH, PS >) = BT >> < G [HTH], PS > .
eq G + (TH1 ; TH2) = G + TH1 ; G + TH2 .

---------------- Typing rules ----------------
op _|-_ : GEnvironment Name ~> Type .
op _|-_ : GEnvironment Process ~> DEnvironment [frozen(2)] .
op _|-_ : GEnvironment System ~> ThInterface [frozen(2)] .

ceq GAMMA . X : T |- P = GAMMA |- P if not X in fn(P) .
ceq GAMMA . X : T |- S = GAMMA |- S if not X in fn(S) .

ceq GAMMA . X : T |- X = T
    if fg(T) subset groups(GAMMA) [label Name] .

eq GAMMA |- 0P = none-D [label NilP] .
ceq GAMMA . X : G[T] |- X (Y : T) . P =
    (GAMMA . X : G[T] . Y : T |- P) + Dr(T)
    if GAMMA . X : G[T] |- X == G[T] [label In] .
ceq GAMMA . X : G[T] |- X < Y >. P =
    (GAMMA . X : G[T] |- P) + Dw(G[T])
    if GAMMA . X : G[T] |- X == G[T] /\ GAMMA |- Y == T [label Out].
eq GAMMA |- (v X : T) P = GAMMA . X : T |- P [label ResNP] .
eq GAMMA |- ! P = GAMMA |- P [label Rep] .
ceq GAMMA |- [[X OP Y]] P = (GAMMA |- Y) OP Y + GAMMA |- P
```

```
        if GAMMA |- Y :: CtxtVar
        /\ GAMMA |- Y == GAMMA |- X [label CondA] .
    eq GAMMA |- [X == Y] (P ; P') =
        (GAMMA |- [[X == Y]]P) + (GAMMA |- [[X =/= Y]]P') [label CondB].
    eq GAMMA |- NEP1 | NEP2 =
        (GAMMA |- NEP1) + (GAMMA |- NEP2) [label ParP] .

    eq GAMMA |- 0S = none-Th [label NilS] .
    eq GAMMA |- (v G) P < U > =
        G [U] + (GAMMA . group: G |- P) [label ResGP] .
    eq GAMMA |- (v R) S = R + (GAMMA . group: R |- S) [label ResGS] .
    eq GAMMA |- (v X : T) S = GAMMA . X : T |- S [label ResNS] .
    eq GAMMA |- NES1 | NES2 =
        (GAMMA |- NES1) ; (GAMMA |- NES2) [label ParS] .

    ---------------- Policy satisfaction ----------------
    op perms : PermissionFunction Hierarchy Set{Group} Purpose
        ~> Set{Permission} .
    op $perms : PermissionFunction Set{Hierarchy} Set{Group} Purpose
        -> Set{Permission} .
    op incompatibleHierarchy : Hierarchy Set{Group}
        -> [Set{Permission}].
    eq perms(PF, nil, GS, U) = empty .
    eq perms(PF, H, empty, U) = empty .
    ceq perms(PF, G : US [HS], GS, U) =
        incompatibleHierarchy(G : US[HS], GS) if not G in GS .
    eq perms(PF, G : US [HS], GS, U) =
        union(if U in US then PF(U, G) else empty fi,
            $perms(PF, if U in US then U -> HS else HS fi,
                intersection(groups(HS), GS), U)) .
    eq $perms(PF, empty, GS, U) = empty .
    eq $perms(PF, (H, HS), GS, U) = union(
        if root(H) subset GS then perms(PF, H, GS, U) else empty fi,
        $perms(PF, HS, GS, U)) .

    op _|=_ : Policy ThInterface -> Bool [prec 50] .
    eq POL |= none-Th = true .
    eq BT >> H, PF |= BT >> < HTH, PS > =
        PS <= perms(PF, H, groups(HTH), purpose(HTH)) .
    eq BT >> H, PF ; POL |= BT >> < HTH, PS > =
        BT >> H, PF |= BT >> < HTH, PS > .
    eq POL |= NETH1 ; NETH2 = POL |= NETH1 and POL |= NETH2 .
    eq POL |= TH = false [owise] .
endfm
```

Βιβλιογραφία

[1] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.

[2] J. Berman and D. Mulligan. Privacy in the digital age: Work in progress. *Nova Law Review*, 23:551, 1998.

[3] J.-W. Byun, E. Bertino, and N. Li. Purpose based access control of complex data for privacy protection. In *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*, pages 102–110. ACM, 2005.

[4] L. Cardelli, G. Ghelli, and A. D. Gordon. Secrecy and group creation. In *International Conference on Concurrency Theory*, pages 365–379. Springer, 2000.

[5] A. Cavoukian. The security-privacy paradox: Issues, misconceptions, and strategies, 2003.

[6] A. Cavoukian. Privacy by design: Origins, meaning, and prospects. *Privacy Protection Measures and Technologies in Business Organizations: Aspects and Standards*, 170, 2011.

[7] M. Clavel, F. Durán, S. Eker, S. Escobar, P. Lincoln, N. Martı-Oliet, J. Meseguer, and C. Talcott. Maude manual (version 2.7). Technical report, Tech. rep., SRI International Computer Science Laboratory, available a t: http://maude. cs. uiuc. edu/maude2-manual, 2015.

[8] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About maude: A High-Performance Logical Framework*. Springer-Verlag, 2007.

[9] R. Cullen. Culture, identity and information privacy in the age of digital government. *Online Information Review*, 33(3):405–421, 2009.

[10] J. DeCew. Privacy. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2015 edition, 2015.

[11] A. Dix. Built-in privacy –no panacea, but a necessary condition for effective privacy protection. *Identity in the Information Society*, 3(2):257–265, 2010.

[12] T. Doyle. Privacy and perfect voyeurism. *Ethics and Information Technology*, 11(3):181–189, 2009.

[13] European Comission. The EU-U.S. privacy shield. `http://ec.europa.eu/justice/data-protection/international-transfers/eu-us-privacy-shield/index_en.htm`.

[14] L. Floridi. Four challenges for a theory of informational privacy. *Ethics and Information Technology*, 8(3):109–119, 2006.

[15] M. Foucault. *Επιτήρηση και τιμωρία*. Εκδόσεις Ράππα, 2004.

[16] S. Jakšic, J. Pantovic, and S. Ghilezan. Linked data privacy. *Mathematical Structures in Computer Science*, pages 1–21, 2015.

[17] E. Kokkinofta and A. Philippou. Type checking purpose-based privacy policies in the π-calculus. In *International Workshop on Web Services and Formal Methods*, pages 122–142. Springer, 2014.

[18] D. Kouzapas and A. Philippou. A methodology for a formal approach in privacy. to appear.

[19] D. Kouzapas and A. Philippou. Type checking privacy policies in the π-calculus. In *International Conference on Formal Techniques for Distributed Objects, Components and Systems*, pages 181–195. Springer, 2015.

[20] G. T. Marx. Privacy and social stratification. *Knowledge, Technology and Policy*, 20(2):91–95, 2007.

[21] T. Mathiesen. The viewer society: Michel Foucault's 'panopticon' revisited. *Theoretical Criminology*, 1(2):215–234, 1997.

[22] J. Meseguer. Twenty years of rewriting logic. *The Journal of Logic and Algebraic Programming*, 81(7):721–781, 2012.

[23] J. Meseguer and G. Roşu. Rewriting logic semantics: From language specifications to formal analysis tools. In *International Joint Conference on Automated Reasoning*, pages 1–44. Springer, 2004.

[24] N. Mooradian. The importance of privacy revisited. *Ethics and Information Technology*, 11(3):163–174, 2009.

[25] Q. Ni, E. Bertino, J. Lobo, C. Brodie, C.-M. Karat, J. Karat, and A. Trombeta. Privacy-aware role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 13(3):24, 2010.

[26] Q. Ni, D. Lin, E. Bertino, and J. Lobo. Conditional privacy-aware role based access control. In *European Symposium on Research in Computer Security*, pages 72–89. Springer, 2007.

[27] G. T. Nojeim. Cybersecurity and freedom on the Internet. *Journal of National Security Law & Policy*, 4:119, 2010.

[28] R. Pardo and G. Schneider. A formal privacy policy framework for social networks. In *International Conference on Software Engineering and Formal Methods*, pages 378–392. Springer, 2014.

[29] J. Parrow. *An Introduction to the π-calculus*, chapter 8, pages 479–543. Elsevier, 2001.

[30] B. C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

[31] J. Rachels. Why privacy is important. *Philosophy & Public Affairs*, pages 323–333, 1975.

[32] S. C. Rickless. The right to privacy unveiled. *San Diego Law Review*, 44(1):773–799, 2007.

[33] A. Rubel. The particularized judgment account of privacy. *Res Publica*, 17(3):275–290, 2011.

[34] D. J. Solove. A taxonomy of privacy. *University of Pennsylvania Law Review*, pages 477–564, 2006.

[35] P. Thati, K. Sen, and N. Martí-Oliet. An executable specification of asynchronous π-calculus semantics and may testing in Maude 2.0. *Electronic Notes in Theoretical Computer Science*, 71:261–281, 2004.

[36] S. D. Warren and L. D. Brandeis. The right to privacy. *Harvard Law Review*, pages 193–220, 1890.

[37] R. Wolff. Emergent privacy. *Journal of Philosophy*, 112(3):141–158, 2015.

[38] Ευρωπαϊκό Κοινοβούλιο and Συμβούλιο της Ευρωπαϊκής Ένωσης. Κανονισμός ΕΕ 2016/679. `http://data.europa.eu/eli/reg/2016/679/oj`.

[39] Μ. Ν. Κουκοβίνη. Εγγενής ενσωμάτωση ιδιωτικότητας σε τεχνολογίες λογισμικού προσανατολισμένου σε υπηρεσίες. Διδακτορική διατριβή, Εθνικό Μετσόβιο Πολυτενχείο, 2014.