Ονοματεπώνυμο: Παντελεήμων Μαλέκας
A.M: 1115201600268

- Πρόβλημα 1

1. Ορίζουμε το πρόβλημα ικανοποίησης περιορισμών χρησιμοποιώντας τα ακόλουθα:

α) Ένα σύνολο μεταβλητών έστω X = $\{X_{11},X_{12},...,X_{ij}\}$ όπου κάθε X αντιστοιχεί στις συντεταγμένες των λευκών τετραγώνων.
β) Ένα πεδίο τιμών το οποίο αντιστοιχεί στους αριθμούς 1 εως 9 για κάθε μεταβλητή. Άρα για κάθε X έχουμε D = {1,2,3,4,5,6,7,8,9}.
γ) Ένα σύνολο περιορισμών το οποίο ορίζει ότι κάθε αλληλουχία των αριθμών πρέπει να αθροίζει στο συγκεκριμένο άθροισμα γραμμής στην γραμμή που βρίσκονται (Π.χ: Αν σε μία γραμμή έχουμε κελί με άθροισμα 4, δύο κενά, και ένα άλλο κελί με άθροισμα αμέσως μετά, οι αριθμοί στα δύο κενά πρέπει να έχουν άθροισμα ίσο με 4. Απο το επόμενο κελί ξεκινά νέο άθροισμα). Αντίστοιχα πρέπει οι αριθμοί σε μία στήλη να έχουν άθροισμα ίσο με το συγκεκριμένο άθροισμα στήλης όπου βρίσκονται. Άρα έχουμε ότι: $R_{ij \rightarrow kl} = X_{ij} + \dots + X_{kl}$ και $C_{ij \rightarrow kl} = X_{ij} + \dots + X_{kl}$ όπου R αντιστοιχεί στον αριθμό αθροίσματος γραμμής και αντίστοιχα C για τις στήλες. Τα ij και kl αντιστοιχούν τις συντεταγμένες από την πρώτη και τελευταία μεταβλητή στο άθροισμα. Επίσης έχουμε ότι για κάθε άθροισμα οι αριθμοί πρέπει να είναι μοναδικοί. Άρα ισχύει ότι: $X_{ij} \neq \dots \neq X_{kl}$ για κάθε μεταβλητή σε ένα άθροισμα.

2. Ο κώδικας που έχω υλοποιήσει είναι στο αρχείο kakuro.py. Χρησιμοποιήθηκαν τα αρχεία csp.py, utils.py και search.py από το Github του βιβλίου, τα οποία δεν τα έχω τροποποίησει (https://github.com/aimacode).
   Εκτελείται ως: python3 kakuro.py difficulty algorithm. Τα ορίσματα θα αναλυθούν στην συνέχεια.
   Συγκεκριμένα υλοποίησα μια κλάση Kakuro που κληρονομεί από την CSP του csp.py. Η κλάση αυτή ορίζει τις μεταβλητές του προβλήματος, τα πεδία τιμών, τους γείτονες από κάθε μεταβλητή και χρησιμοποιεί μια συνάρτηση που ελέγχει για κάθε ανάθεση αν ικανοποιούνται οι περιορισμοί του προβλήματος. Επίσης έχουν οριστεί κάποιες επιπλέον δομές που αποθηκεύουν πληροφορίες που αφορούν το άθροισμα για κάθε γραμμή και στήλη.
   Ακολουθεί η main συνάρτηση όπου ελέγχεται το πρώτο όρισμα της γραμμής εντολών, το difficulty. Ο χρήστης μπορεί να επιλέξει ανάμεσα σε τρεις βαθμούς δυσκολίας: easy, medium και hard. Οι πίνακες που αντιστοιχούν σε κάθε δυσκολία βρίσκονται στην αρχή του αρχείου. Αφού λοιπόν εκτυπωθεί ο αρχικός πίνακας με τα κενά, αρχικοποιείται ένα αντικείμενο της κλάσης Kakuro με βάση τον πίνακα. Στην συνέχεια ακολουθεί η εκτέλεση του αλγορίθμου που επιλέχθηκε από τον χρήστη, όπου ο κάθε αλγόριθμος δίνεται από το δεύτερο όρισμα της γραμμής εντολών. Υπάρχουν έξι επιλογές: BT, BT+LCV, FC, FC+LCV, MAC και MAC+LCV. Οι αλγόριθμοι έχουν χρησιμοποιηθεί από το αρχείο csp.py. Το κριτήριο επιλογής ήταν να εξεταστεί η χρήση των τεχνικών διάδοσης περιορισμών καθώς και των βοηθητικών ευρετικών συναρτήσεων όπως του LCV. Αφού εκτελεστεί ο αλγόριθμος που επιλέχθηκε, εκτυπώνεται ο πίνακας με τις τιμές που

αναθέσαμε σε κάθε μεταβλητή καθώς και ο συνολικός χρόνος εκτέλεσης του αλγορίθμου.

3. Ακολουθεί πίνακας όπου για κάθε πίνακα δείχνει ενδεικτικούς χρόνους εκτέλεσης για κάθε αλγόριθμο (σε δευτερόλεπτα):

	BT	**BT+LCV**	**FC**	**FC+LCV**	**MAC**	**MAC+LCV**
easy	0.010573	0.021457	0.019378	0.029896	0.003578	0.004219
medium	0.000830	0.001547	0.0014740	0.002224	0.00266	0.003031
hard	27.793987	57.88327	55.980336	75.612774	36.758202	45.073735

Παρατηρούμε ότι: Γενικά οι αλγοριθμοι BT και MAC είχαν ανάλογα αποτελέσματα ένω και στις τρεις περιπτώσεις ο FC αποδείχθηκε ο πιο αργός αλγόριθμος. Συγκεκριμένα ο MAC παρείχε την καλύτερη απόδοση στον easy πίνακα ενώ ήταν λίγο πιο αργός στους medium και hard.

Η χρήση του LCV αποδείχθηκε χρονοβόρα για κάθε αλγόριθμο το οποίο ήταν αναμενόμενο καθώς γίνεται ο έλεγχος των περιορισμών και όταν καλείται η συνάρτηση lcv αλλά και μετά από την κλήση της lcv.

Ο αλγόριθμος FC όπως ξέρουμε ελέγχει για κάθε ανάθεση τα πεδία τιμών από τις γειτονικές μεταβλητές και κλαδεύει τις τιμές που δεν είναι συνεπείς με την συγκεκριμένη ανάθεση. Αυτή η διαδικασία αποδείχθηκε αρκετά χρονοβόρα, είδικα στον hard πίνακα, για αυτό και ο FC ήταν ο πιο αργός αλγόριθμος.

Ο MAC αλγόριθμος εντοπίζει έγκαιρα τις ακμές του γράφου αναζήτησης που δεν είναι συνεπείς, δηλαδή τις αναθέσεις που δεν προσφέρουν κάτι ως προς την λύση του προβλήματος μας. Ο AC3b αλγόριθμος που χρησιμοποιεί η mac συνάρτηση αποδείχθηκε αποδοτικός προσφέροντας παρόμοια αποτελέσματα για τους medium και hard πίνακες ενώ στον easy ήταν ο πιο γρήγορος.

(Λεπτομέρειες εκτέλεσης: Τα συγκεκριμένα αποτελέσματα εξετάστηκαν σε περιβάλλον: Ubuntu 18.04.2 και η έκδοση της Python που χρησιμοποιήθηκε είναι: Python 3.6.9)

- Πρόβλημα 2

1. Ορίζουμε το πρόβλημα ικανοποίησης περιορισμών χρησιμοποιώντας τα ακόλουθα:
α) Ένα σύνολο μεταβλητών, έστω: $X_{αδ}$ τον χρόνο από την αίθουσα προς τα δωμάτια και $X_{δα}$ αντίστροφα, $X_{ξ}$ ως ο χρόνος ξεκούρασης, $X_{αχρ}$ τον χρόνο για να πάει κανείς προς το χρηματοκιβώτιο. Αντίστοιχα $X_{χρα}$ τον χρόνο επιστροφής προς την αίθουσα. Τέλος ορίζουμε $X_{π}$

τον χρόνο της παραβίασης του χρηματοκιβωτίου. Να σημειωθεί ότι κάθε μεταβλητή είναι ξεχωριστή για κάθε ύποπτο. Δηλαδή έχουμε π.χ. $X_{αδ1}$ για τον κ. Γιάννη, $X_{αδ2}$ για την κ. Μαρία κ.τ.λ.
β) Ένα πεδίο πιθανών τιμών το οποίο αντιστοιχεί στα λεπτά της ώρας για την κάθε μεταβλητή.
γ) Ένα σύνολο περιορισμών το οποίο ορίζει τα λεπτά που μπορούν να ανατεθούν σε κάθε μεταβλητή. Συγκεκριμένα: $5 \leq X_{δα}, X_{αδ} \leq 10$ και $20 \leq X_{αχρ}, X_{χρα} \leq 30$ και $45 \leq X_{π} \leq 90$. Επίσης ορίζεται ο χρόνος ολοκλήρωσης ομιλίας ως 9:30 για τον κύριο Γιάννη, 10:00 για την κυρία Μαρία και 10:30 για την κυρία Όλγα καθώς και τον χρόνο απονομής του βραβείου στις 11:00.

2. Ο αστυνόμος Σιεσπής συνέλαβε τον κύριο Γιάννη. Ακολουθεί εξήγηση:

Αρχικά έχουμε ότι ο ελάχιστος χρόνος που χρειάζεται κανείς για να κλέψει το έπαθλο είναι 85 λεπτά. Στην συνάρτηση $X_{αχρ} + X_{π} + X_{χρα}$, αν θέσουμε τις ελάχιστες τιμές στις μεταβλητές θα δούμε ότι έχουμε 85. Επίσης δεν υφίσταται το ενδεχόμενο κάποιος και να επέστρεψε σπίτι του και να έκλεψε το έπαθλο. Συνεπώς κάποιος είπε ψέματα.

Οπότε στην κυρία Όλγα, έχουμε ότι: αν θεωρήσουμε ότι είναι ένοχη χρειάζεται τουλάχιστον 85 λεπτά από τις 10:30 για να κλέψει το έπαθλο. Από την στιγμή που ήταν όμως στην απονομή δεν είχε τον απαραίτητο χρόνο. Άρα πράγματι γύρισε σπίτι της, οπότε θέτουμε τις κατάλληλες τιμές στις μεταβλητές $X_{αδ3}$, $X_{ξ3}$ και $X_{δα3}$ έχοντας ότι $X_{αδ3} + X_{ξ1} + X_{δα3} = 30$, μιας και αυτόν τον χρόνο είχε ελεύθερο στην διάθεση της η κ. Όλγα μέχρι την απονομή. Οι υπόλοιπες μεταβλητές δεν παίρνουν κάποια τιμή.

Η κυρία Μαρία επίσης δεν είναι ένοχη επειδή τελείωσε την ομιλία της στις 10:00 και μέχρι την απονομή δεν είχε 85 λεπτά για να κλέψει το έπαθλο. Οπότε θέτουμε όπως και πριν τις τιμές που χρειάζονται οι μεταβλητες $X_{αδ2}$, $X_{ξ2}$ και $X_{δα2}$ έτσι ώστε $X_{αδ2} + X_{ξ2} + X_{δα2} = 60$ όπου αυτός ήταν ο ελεύθερος χρόνος της κ. Μαρίας.

Φτάνουμε λοιπόν στον κύριο Γιάννη ο οποίος τελείωσε την ομιλία του στις 9:30. Βλέπουμε ότι αν προσθέσουμε 85 λεπτά φτάνουμε στις 10:55. Εφόσον δεν υπάρχει άλλος ύποπτος βρίσκουμε ότι ο κύριος Γιάννης έκλεψε το έπαθλο. Οπότε θέτουμε τις απαραίτητες τιμές στις μεταβλητές $X_{αχρ1}$, $X_{π1}$ και $X_{χρα1}$ έτσι ώστε $X_{αχρ1} + X_{π1} + X_{χρα1} = 90$ μιας και αυτός είναι ο χρόνος που έχει ο κύριος Γιάννης από την στιγμή που τελείωσε η ομιλία του μέχρι να παραβιάσει το χρηματοκιβώτιο και να επιστρέψει στην απονομή. Οι υπόλοιπες μεταβλητές δεν παίρνουν κάποια τιμή.

3. Μία μέθοδος διάδοσης περιορισμών είναι η χρήση του αλγορίθμου Forward Checking. Η λογική είναι ότι στα σημεία όπου θέτουμε τιμές στις μεταβλητές από τις συναρτήσεις $X_{αδ} + X_{ξ} + X_{δα}$ και $X_{αχρ} + X_{π} + X_{χρα}$, οταν θέσουμε μία τιμή ελέγχουμε τις υπόλοιπες μεταβλητές όπου δεν έχουν πάρει τιμές και διαγράφουμε τις τιμές που είναι ασυνεπείς με την προηγούμενη ανάθεση. Π.χ: Για την κ. Όλγα έχουμε την συνάρτηση $X_{αδ3} + X_{ξ3} + X_{δα3} = 30$. Αν θέσουμε στην $X_{αδ3}$ την τιμη 10 και στην $X_{ξ3}$ την τιμη 15, τότε για την μεταβλητή $X_{δα3}$ μας μένει μόνο η τιμή 5 επειδή οποιαδήποτε μεγαλύτερη τιμή θα βγάλει άθροισμα πάνω από 30. Οπότε σε αυτή την περίπτωση διαγράφουμε τις τιμές πάνω από 5 όταν θα χρειαστεί να θέσουμε τιμή στην μεταβλητή $X_{δα3}$.

- Πρόβλημα 3

1. Ορίζουμε το πρόβλημα ικανοποίησης περιορισμών χρησιμοποιώντας τα ακόλουθα:

α) Ένα σύνολο μεταβλητών $X = \{X_{11},X_{12},...,X_{1m},X_{21},...,X_{nm}\}$ όπου n οι εργασίες και m οι ενέργειες για κάθε εργασία.

β) Ένα πεδίο (m,d_i) που αποτελείται από: ένα σύνολο πιθανών τιμών m το οποίο αντιστοιχεί στον αριθμό των μηχανών και d_i την διάρκεια από κάθε ενέργεια. Οπότε σε κάθε ενέργεια θέτουμε τον αριθμό της μηχανής στην οποία εκτελείται και την χρονική της διάρκεια.

γ) Οι περιορισμοί που έχουμε είναι: Κάθε ενέργεια μπορεί να εκτελεστεί αποκλειστικά σε μία από τις διαθέσιμες μηχανές άρα το m είναι μοναδικό για κάθε ενέργεια X. Μία ενέργεια, έστω X_{ij} μπορεί να εκτελεστεί αφού ολοκληρωθούν οι X_{i1} μεχρι X_{ij-1} ενέργειες, δηλαδή όλες οι προηγούμενες. Όταν ξεκινήσει η εκτέλεση μιας εντολής δεν μπορεί να διακοπεί αρά κάθε μηχανή m είναι κατειλημμένη μέχρι την ολοκλήρωση μιας ενέργειας, οπότε το m_i δεν μπορεί να ανατεθεί σε μία ενέργεια X_{ij} ενεργεία όσο εκτελείται κάποια άλλη σε αυτή. Τέλος, εφόσον κάθε εργασία πρέπει να ολοκληρωθεί πριν την προθεσμία D>0 πρέπει $d_1 + d_2 + \dots + d_m \leq D$, όπου d η διάρκεια από την κάθε ενέργεια.

2. Αν θεωρήσουμε ότι ο χρόνος d_i για κάθε ενέργεια είναι ίδιος και ότι το D είναι μεγαλύτερο από το άθροισμα όλων των d_i, τότε για n=3 και m=4 μία λύση είναι να θέσουμε την κάθε πρώτη ενέργεια από μια εργασία σε μια διαφορετική μηχανή. Αυτό θα έχει ως αποτέλεσμα να εκτελεστούν οι ενέργειες π.χ. της 1ης εργασίας στην πρώτη μηχανή, οι ενέργειες από την 2η εργασία στην δεύτερη μηχανή και το ίδιο με την 3η εργασία. Εφόσον ο χρόνος για κάθε ενέργεια είναι ίδιος αυτό θα έχει ως αποτέλεσμα να εκτελεστούν και παράλληλα οι 3 εργασίες.

   Ένα παράδειγμα το οποίο είναι μη συνεπές είναι η περίπτωση όπου το άθροισμα των d_i τιμών θα ήταν μεγαλύτερο από την σταθερά D. Οπότε για παράδειγμα με n=3 και m=4, αν το D είχε τιμή 12 και κάθε μεταβλητή-ενέργεια έπαιρνε ως d_i το 1 με εξαίρεση την τελευταία να έχει 2 αυτό θα είχε ως αποτέλεσμα το άθροισμα των d_i να είναι 13, δηλαδή μεγαλύτερο του D.

3. Ένας αλγόριθμος υπαναχώρησης που θα μπορούσε να χρησιμοποιήθει είναι ο MAC (Maintaining Arc Consistency). Ένα βασικό χαρακτηριστικό που κάνει τους αλγόριθμους υπαναχώρησης πιο αποδοτικούς από τον βασικό Backtracking αλγόριθμο είναι όταν εφαρμόζουν κάποια τεχνική διάδοσης περιορισμών. Αυτές οι τεχνικές μας βοηθάνε π.χ. στο να κλαδεύουμε τμήματα του χώρου αναζήτησης έτσι ώστε να βρίσκουμε λύση για ένα CSP πρόβλημα πιο αποδοτικά. Ο MAC είναι ένας τέτοιος αλγόριθμος και συγκεκριμένα μπορεί και εντοπίζει έγκαιρα τις ακμές του γράφου αναζήτησης που δεν είναι συνεπείς, δηλαδή τις αναθέσεις που δεν προσφέρουν κάτι ως προς την λύση ενός CSP προβλήματος.

- Πρόβλημα 4

α) **(A ^ B ^ C ⇒ D) ⇔ (A ⇒ (B ⇒ (C ⇒ D)))**
Έστω για αυτήν την πρόταση ότι P = (A ^ B ^ C ⇒ D) και R = (A ⇒ (B ⇒ (C ⇒ D)))

Ακολουθεί ο πίνακας αλήθειας της πρότασης:

A	**B**	**C**	**D**	**A^B**	**A^B^C**	**C⇒D**	**A^B^C⇒D**	**B⇒(C⇒D)**	**A⇒(B⇒(C⇒D))**	**P⇒R**	**R⇒P**	**P⇔R**
F	F	F	F	F	F	T	T	T	T	T	T	T
F	F	F	T	F	F	T	T	T	T	T	T	T
F	F	T	F	F	F	F	T	T	T	T	T	T
F	F	T	T	F	F	T	T	T	T	T	T	T
F	T	F	F	F	F	T	T	T	T	T	T	T
F	T	F	T	F	F	T	T	T	T	T	T	T
F	T	T	F	F	F	F	T	F	T	T	T	T
F	T	T	T	F	F	T	T	T	T	T	T	T
T	F	F	F	F	F	T	T	T	T	T	T	T
T	F	F	T	F	F	T	T	T	T	T	T	T
T	F	T	F	F	F	F	T	T	T	T	T	T
T	F	T	T	F	F	T	T	T	T	T	T	T
T	T	F	F	T	F	T	T	T	T	T	T	T
T	T	F	T	T	F	T	T	T	T	T	T	T
T	T	T	F	T	T	F	F	F	F	T	T	T
T	T	T	T	T	T	T	T	T	T	T	T	T

β) **A^(A⇒B)^(A⇒¬B)**

Ακολουθεί ο πίνακας αλήθειας της πρότασης:

A	**B**	**¬B**	**A⇒B**	**A⇒¬B**	**A^(A⇒B)**	**A^(A⇒B)^(A⇒¬B)**
F	F	T	T	T	F	F
F	T	F	T	T	F	F
T	F	T	F	T	F	F
T	T	F	T	F	T	F

γ) **(A∨B)^(¬A∨C)^¬B^¬C**

Ακολουθεί ο πίνακας αλήθειας της πρότασης:

A	**B**	**C**	**¬A**	**¬B**	**¬C**	**A∨B**	**¬A∨C**	**(A∨B)^(¬A∨C)**	**(A∨B)^(¬A∨C)^¬B**	**(A∨B)^(¬A∨C)^¬B^¬C**
F	F	F	T	T	T	F	T	F	F	F
F	F	T	T	T	F	F	T	F	F	F
F	T	F	T	F	T	T	T	T	F	F
F	T	T	T	F	F	T	T	T	F	F
T	F	F	F	T	T	T	F	F	F	F
T	F	T	F	T	F	T	T	T	T	F
T	T	F	F	F	T	T	F	F	F	F
T	T	T	F	F	T	T	T	T	F	F

δ) **(A∨B)^(¬A∨C)^(B∨C)**

Ακολουθεί ο πίνακας αλήθειας της πρότασης:

A	**B**	**C**	**¬A**	**A∨B**	**¬A∨C**	**B∨C**	**(A∨B)^(¬A∨C)**	**(A∨B)^(¬A∨C)^(B∨C)**
F	F	F	T	F	T	F	F	F
F	F	T	T	F	T	T	F	F
F	T	F	T	T	T	T	T	T
F	T	T	T	T	T	T	T	T
T	F	F	F	T	F	F	F	F
T	F	T	F	T	T	T	T	T
T	T	F	F	T	F	T	F	F
T	T	T	F	T	T	T	T	T

Άρα έχουμε ότι:

1. Η πρόταση α) είναι η μοναδική έγκυρη πρόταση εφόσον είναι αληθής για κάθε πιθανή ερμηνεία.
2. Οι προτάσεις α) και δ) είναι ικανοποιήσιμες επειδή έχουν τουλάχιστον μία ερμηνεία που είναι αληθής.
3. Οι προτάσεις β) και γ) είναι μη ικανοποιήσιμες γιατί δεν έχουν καμία ερμηνεία που να είναι αληθής.
4. Οι προτάσεις α) και δ) έχουν τουλάχιστον ένα μοντέλο γιατί έχουν από τουλάχιστον μία αληθή ερμηνεία.
5. Η πρόταση α) είναι η μοναδική ταυτολογία που έχουμε γιατί όπως είδαμε προηγουμένως είναι μία έγκυρη πρόταση και οι έγκυρες προτάσεις λέγονται και ταυτολογίες.
6. Καμία πρόταση δεν είναι σε μορφή Horn. Σύμφωνα με τις διαφάνειες του μαθήματος μία φράση Horn είναι μία φράση με το πολύ ένα θετικό λεκτικό. Όμως εδώ οι προτάσεις που μας δίνονται έχουν πάνω από ένα θετικό λεκτικό. Για να ήταν σε μορφή Horn θα έπρεπε να είχαν ή ένα ή κανένα θετικό λεκτικό.

- Πρόβλημα 5

Ορίζουμε ως KB (από τα αρχικά του Knowledge Base) την πρόταση: A ^ (B ⇔ C) και ως φ την πρόταση (A ^ B) ⇔ (A ^ C). Για να αποδείξουμε ότι KB ⊨ φ, δηλαδή ότι η KB καλύπτει λογικά την φ χρησιμοποιώντας ανάλυση, θα εφαρμόσουμε τον κανόνα της ανάλυσης στην πρόταση KB ^ ¬φ. Αν η KB καλύπτει λογικά την φ, τότε πρέπει η KB ^ ¬φ να είναι ψευδής για κάθε πιθανή ερμηνεία.

Οπότε παίρνουμε την πρόταση: A ^ (B ⇔ C) ^ ¬((A ^ B) ⇔ (A ^ C)) και την μετατρέπουμε σε προτάσεις CNF. Άρα έχουμε τις προτάσεις: A, (¬B V C), (B V ¬C), (A V ¬A V ¬B), (B V ¬A V ¬B), (¬A V ¬C V ¬A V ¬B), (A V A), (B V A), (¬A V ¬C V A), (A V C), (B V C) και (¬A V ¬C V C).

Αρχίζουμε λοιπόν να εφαρμόζουμε τον κανόνα της ανάλυσης. Επιλέγουμε το A και βλέπουμε ότι επειδή είναι η πρώτη πρόταση που έχουμε, δεν παράγει κάποια νέα πρόταση.
Οι προτάσεις (¬B V C) και (B V ¬C) επίσης δεν παράγουν κάποια καινούργια. Η πρόταση (A V ¬A V ¬B) διαγράφεται γιατί κάθε συνδυασμός με τις προηγούμενες προτάσεις την εξαλείφει.
Από την (B V ¬A V ¬B) παράγεται η (¬A V ¬B V C) από την (¬B V C) και η (B V ¬A V ¬C) από την (B V ¬C). Κρατάμε λοιπόν το πρώτο νέο σετ προτάσεων: ((¬A V ¬B V C), (B V ¬A V ¬C)).
Στην συνέχεια, από την (¬A V ¬C V ¬A V ¬B) παράγεται η (¬C V ¬A V ¬B) και η (¬A V ¬C V ¬B) από την A. Επίσης παράγεται η (¬A V ¬B) από την (¬B V C), η (¬A V ¬C) από την (B V ¬C) και η (¬A V ¬C V ¬B) από την (B V ¬A V ¬B). Δεύτερο νέο σετ: ((¬C V ¬A V ¬B), (¬A V ¬C V ¬B), (¬A V ¬B), (¬A V ¬C), (¬A V ¬C V ¬B)).
Παρακάτω, οι άλλες προτάσεις που έχουμε διαγράφονται μέχρι που φτάνουμε στην (B V C) από την οποία παράγονται: η C από την (¬B V C), η B από την (B V ¬C) και η (C V B V ¬A) από την (B V ¬A V ¬B). Τρίτο νέο σετ: (C, B, (C V B V ¬A)).
Παρακάτω, από την (¬A V ¬C V C) παράγονται: η (¬A V C V ¬B) από την (¬B V C), η (¬A V ¬C V B) από την (B V ¬C), η (¬A V ¬C V ¬B) από την (¬A V ¬C V ¬A V ¬B) και η (¬A V C V B) από την (B C). Τέταρτο νέο σετ: ((¬A V C V ¬B), (¬A V ¬C V B), (¬A V ¬C V ¬B), (¬A V C V B)).
Φτάνουμε στην (¬C V ¬A V ¬B) από την παράγονται: η (¬C V ¬B) από την A, η (¬A V ¬B) από την (¬B V C), η (¬C V ¬A) από την (B V ¬C), η (¬C V ¬A V ¬B) από την (B V ¬A V ¬B) και η (¬A V ¬B V ¬C) από την (¬A V ¬C V C). Πέμπτο νέο σετ: ((¬C V ¬B), (¬A V ¬B), (¬C V ¬A), (¬C V ¬A V ¬B), (¬A V ¬B V ¬C)).
Προχωράμε στην (¬A V ¬B) από την οποία παράγονται: η ¬B από την A, η (¬A V ¬C) από την (B V ¬C), η (¬A V ¬B) από την (B V ¬A V ¬B) και η (¬A V C) από την (B V C). Έκτο νέο σετ: (¬B, (¬A V ¬C), (¬A V ¬B), (¬A V C)).
Παρακάτω, από την (¬A V ¬C) παράγονται: η ¬C από την A, η (¬A V ¬B) από την (¬B V C), η (¬A V B) από την (B V C) και η (¬A V ¬C) από την (¬A V ¬C V C). Έβδομο νέο σετ: (¬C,(¬A V ¬B), (¬A V B), (¬A V ¬C)).

Από την C παράγονται: η B από την (B V ¬C), η (¬A V ¬B) από την (¬A V ¬C V ¬A V ¬B), η (¬A V C) από την (¬A V ¬C V C), η (¬A V ¬B) από την (¬C V ¬A V ¬B) και η ¬A από την (¬A V ¬C). Όγδοο νέο σετ: (B, (¬A V ¬B), (¬A V C), (¬A V ¬B), ¬A)
Επιλέγουμε την B από την οποία παράγονται: η C από την (¬B V C), η (B V ¬A) από την (B V ¬A V ¬B), η (¬A V ¬C) από την (¬A V ¬C V ¬A V ¬B), η (¬C V ¬A) από την (¬C V ¬A V ¬B) και η ¬A απο την ¬A V ¬B. Ενατο νεο σετ: (C, (B V ¬A), (¬A V ¬C), (¬C V ¬A), ¬A)
Προχωράμε στην (¬C V ¬B) από την οποία παράγονται: η ¬B από την (¬B V C), η ¬C από την (B V ¬C), η (¬C V ¬A V ¬B) από την (B V ¬A V ¬B), η (¬B V ¬A V ¬C) από την (¬A V ¬C V C), η ¬B από την C και η ¬C από την B. Δέκατο νεο σετ: (¬B, ¬C, (¬C V ¬A V ¬B), (¬B V ¬A V ¬C), ¬B, ¬C).
Τέλος, επιλέγουμε την ¬B όπου: παράγεται η ¬C από την (B V ¬C), η (¬A V ¬B) από την (B V ¬A V ¬B), η C από την (B V C) και τελικά φτάνουμε στην B από την οποία έχουμε την κενή φράση. Εφόσον λοιπόν έχουμε μια κενή φράση, άρα μία πρόταση που δεν ικανοποιείται από κανένα μοντέλο, καταλήγουμε πως ισχύει ότι: KB ⊨ φ άρα η A ^ (B ⇔ C) καλύπτει λογικά την (A ^ B) ⇔ (A ^ C).

- Πρόβλημα 6

Σύμφωνα με τις διαφάνειες του μαθήματος έχουμε ότι:

Η παρακάτω γραμματική χωρίς συμφραζόμενα ορίζει τις **καλά ορισμένες προτάσεις (well-formed sentences)** της προτασιακής λογικής:

$$Sentence \rightarrow AtomicSentence \mid ComplexSentence$$
$$AtomicSentence \rightarrow \mathbf{True} \mid \mathbf{False} \mid Symbol$$
$$Symbol \rightarrow \mathbf{P_1} \mid \mathbf{P_2} \mid \cdots$$
$$ComplexSentence \rightarrow (Sentence) \mid \neg Sentence$$
$$\mid Sentence\ BinaryConnective\ Sentence$$
$$BinaryConnective \rightarrow \wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow$$

Οπότε με βάση αυτά, απαντάμε:

- (A)

Είναι καλά ορισμένη πρόταση της προτασιακής λογικής και συγκεκριμένα υπάγεται στην μορφή των Complex Sentences.

- (A → B)

Δεν είναι καλά ορισμένη πρόταση της προτασιακής λογικής. Παρόλο που το σύμβολο → φαίνεται ως έγκυρο σύμβολο, σύμφωνα με τις διαφάνειες έχουμε ότι χρησιμοποιείται το ⇒ ως το σύμβολο για την συνεπαγωγή. Για αυτό η συγκεκριμένη πρόταση δεν θεωρείται καλά ορισμένη.

- A ≡ B

Δεν είναι καλά ορισμένη πρόταση. Το σύμβολο ≡ δεν ανήκει στα σύμβολα των καλά ορισμένων προτάσεων. Για την ισοδυναμία χρησιμοποιείται το ⇔.

- A ⊨ B

Δεν είναι καλά ορισμένη πρόταση. Το σύμβολο ⊨ της λογικής κάλυψης δεν ανήκει στα σύμβολα των καλά ορισμένων λογικών προτάσεων.

- (A ⋀ 1)

Δεν είναι καλά ορισμένη πρόταση. Το 1 δεν είναι σύμβολο της προτασιακής λογικής.