Αναπαράσταση Γνώσης με Λογική

Προτασιακή Λογική

Στόχοι Διάλεξης

- Εισαγωγή στην έννοια της Αναπαράστασης Γνώσης και κατανόηση της δομής ενός πράκτορα που βασίζεται στη γνώση
  - Η λογική ως βασική γλώσσα αναπαράστασης γνώσης και η χρήση της για μοντελοποίηση πληροφοριών από πράκτορες
- Εισαγωγή στην Προτασιακή Λογική
  - Περιγραφή βασικών εννοιών και μεθόδων εξαγωγής γνώσης από υπάρχουσα

Περιεχόμενα

- **Πράκτορες Βασισμένοι στη Γνώση (Knowledge-based agents)**
- Ένα παράδειγμα: **Wumpus world**
- **Γενικά για Λογική**
- **Προτασιακή (Boolean) λογική**
  - Μοντέλα, ισοδυναμία, εγκυρότητα, ικανοποιησιμότητα
  - Κανόνες εξαγωγής συμπερασμάτων και απόδειξη θεωρημάτων
    - forward chaining
    - backward chaining
    - resolution
  - Αποδοτικές μέθοδοι
- **Λογική Πρώτης Τάξης (first-order logic)**

Πράκτορες Βασισμένοι στη Γνώση

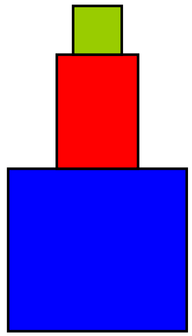
- **Οι πράκτορες που βασίζονται στη γνώση** (knowledge-based agents) μπορούν να γίνουν αντιληπτοί ως πράκτορες που **ξέρουν** πράγματα σχετικά με το περιβάλλον και **συλλογίζονται** σχετικά με τις ενέργειες που θα εκτελέσουν
- **Βασικά συστατικά:**
  - Η **Βάση Γνώσης** (Knowledge Base) είναι ένα σύνολο προτάσεων σχετικά με τον κόσμο σε κάποια επίσημη γλώσσα
  - Η **Γλώσσα Αναπαράστασης Γνώσης** (Knowledge Representation Language) είναι μια γλώσσα της οποίας οι προτάσεις αντιπροσωπεύουν γεγονότα σχετικά με τον κόσμο

Πράκτορες Βασισμένοι στη Γνώση

- **Βασικά συστατικά:**
  - Λειτουργίες για
    - **Προσθήκη πληροφοριών** (προτάσεων) στη βάση γνώσης
      - TELL
    - **Ερωτήσεις** σχετικά με το τι είναι γνωστό
      - ASK
    - Παρόμοια με updates και queries στις βάσεις δεδομένων

  - Ο **μηχανισμός εξαγωγής συμπερασμάτων** (inference mechanism) είναι ένας μηχανισμός που προσδιορίζει τι **συνεπάγεται** από την πληροφορία που έχει προστεθεί στη βάση γνώσης
    - Οι ερωτήσεις που γίνονται στη βάση χρησιμοποιούν αυτό τον μηχανισμό

Παράδειγμα Εξαγωγής Συμπερασμάτων



- **Δεδομένων:**
  - “Ο κόκκινος κύβος είναι πάνω από τον μπλε κύβο”
  - “Ο πράσινος κύβος είναι πάνω από τον κόκκινο κύβο”
- **Συμπέρανε:**
  - “Ο πράσινος κύβος είναι πάνω από τον μπλε κύβο”
  - “Οι κύβοι σχηματίζουν έναν πύργο”

Παράδειγμα Εξαγωγής Συμπερασμάτων

- **Δεδομένων:**

  Αν έχει ηλιοφάνεια σήμερα, τότε ο ήλιος λάμπει στην οθόνη. Αν ο ήλιος λάμψει στην οθόνη, τότε οι κουρτίνες πρέπει να τραβηχτούν. Οι κουρτίνες δεν είναι τραβηγμένες.

- **Απάντησε:**

  Έχει ηλιοφάνεια σήμερα?

Πράκτορες Βασισμένοι στη Γνώση

- Μπορούμε να περιγράψουμε έναν πράκτορα που βασίζεται στη γνώση σε τρία επίπεδα:
  - Το **επίπεδο γνώσης** ή **επιστημολογικό επίπεδο**
    - Σε αυτό το επίπεδο μπορούμε να περιγράψουμε τον πράκτορα προσδιορίζοντας τι ξέρει για τον κόσμο
  - Το **λογικό επίπεδο**
    - Σε αυτό το επίπεδο η γνώση κωδικοποιείται σε προτάσεις κάποιας γλώσσας αναπαράστασης γνώσης
  - Το **επίπεδο υλοποίησης**
    - Σε αυτό το επίπεδο οι προτάσεις υλοποιούνται (π.χ. με μια γλώσσα προγραμματισμού όπως η Prolog)

Πράκτορες Βασισμένοι στη Γνώση

- Παράδειγμα:

  - **Επίπεδο γνώσης** ή **επιστημολογικό επίπεδο**
    - Ο αυτοματοποιημένος οδηγός ταξί ξέρει ότι το Ρίο και το Αντίρριο συνδέονται με γέφυρα
  - **Λογικό επίπεδο**
    - Ο αυτοματοποιημένος οδηγός ταξί έχει την πρόταση (σε λογική πρώτης τάξης) *Συνδέει (Γέφυρα, Ρίο, Αντίρριο)*
  - Το **επίπεδο υλοποίησης**
    - Σε αυτό το επίπεδο η πρόταση *Συνδέει (Γέφυρα, Ρίο, Αντίρριο)* υλοποιείται με μια γλώσσα προγραμματισμού (π.χ. Prolog)

Πράκτορες Βασισμένοι στη Γνώση

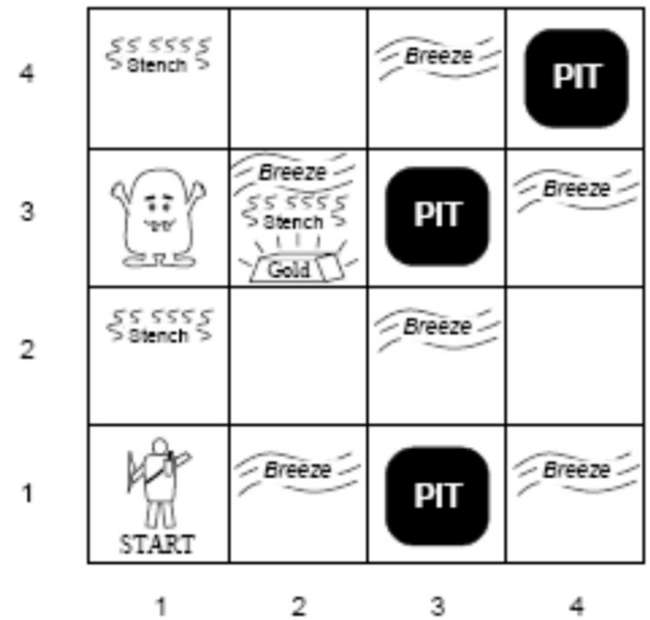
```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

- Ο πράκτορας πρέπει να:
    - Αναπαριστά καταστάσεις, ενέργειες, κτλ.
    - Εισάγει νέες αντιλήψεις
    - Ανανεώνει την εσωτερική αναπαράσταση του κόσμου
    - Εξάγει κρυφές ιδιότητες του κόσμου
    - Εξάγει κατάλληλες ενέργειες

Ο κόσμος του Wumpus

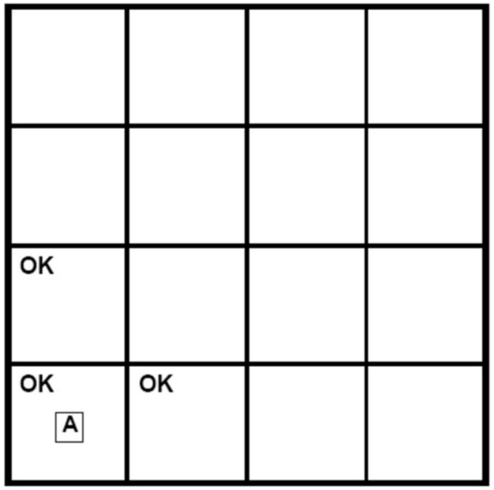
- Μέτρο απόδοσης:
  - +1000 για χρυσό
  - -1000 για wumpus, παγίδα (γούβα)
  - -1 για κάθε βήμα
  - -10 για βέλος
- Περιβάλλον:
  - Πιθανότητα 20% για παγίδα
  - Σε Τετράγωνα δίπλα στο Wumpus υπάρχει δυσοσμία και σε τετράγωνο δίπλα σε παγίδα υπάρχει αύρα, στο τετράγωνο όπου είναι ο χρυσός υπάρχει μια λάμψη, κτλ.
- Μηχανισμοί δράσης
  - Μετακίνηση
  - Στροφή 90°
  - Αρπαγή
  - Εξακόντιση
- Αντιλήψεις
  - [Δυσοσμία, Αύρα, Λάμψη, Γδούπος, Κραυγή]

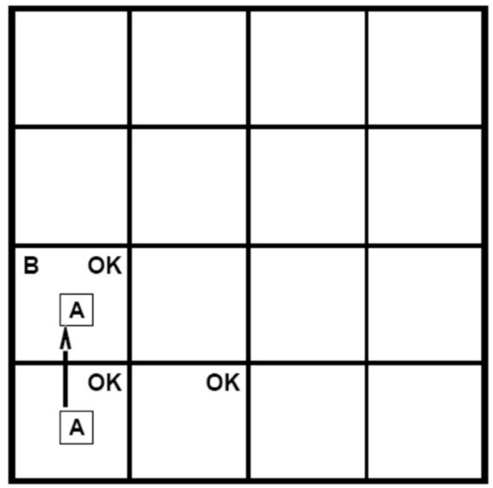


Το περιβάλλον του Wumpus

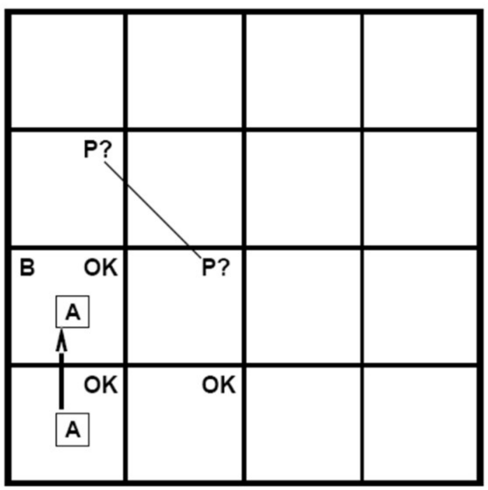
- **Πλήρως παρατηρήσιμο?**
  - Όχι – μόνο τοπική παρατήρηση
- **Αιτιοκρατικό?**
  - Ναι – τα αποτελέσματα των ενεργειών είναι προκαθορισμένα
- **Επεισοδιακό?**
  - Όχι – ακολουθιακό σε ότι αφορά τις ενέργειες
- **Στατικό?**
  - Ναι – το Wumpus και οι παγίδες (pits) δεν μετακινούνται
- **Διακριτό?**
  - Ναι
- **Μονοπρακτορικό?**
  - Ναι – το Wumpus είναι μέρος του περιβάλλοντος, δεν είναι πράκτορας

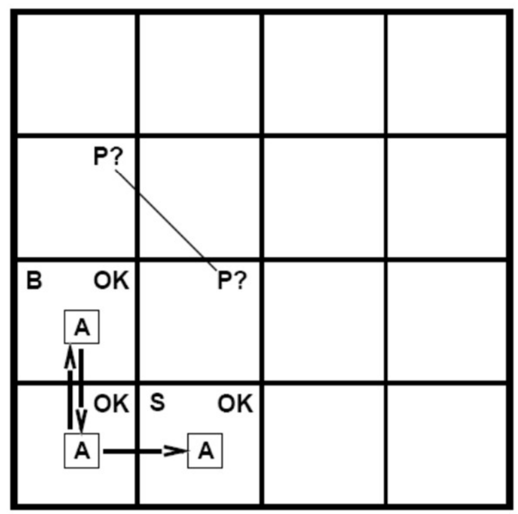
Εξερεύνηση στον κόσμο του Wumpus



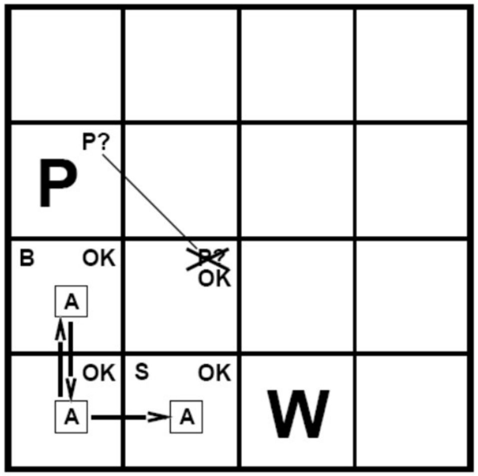


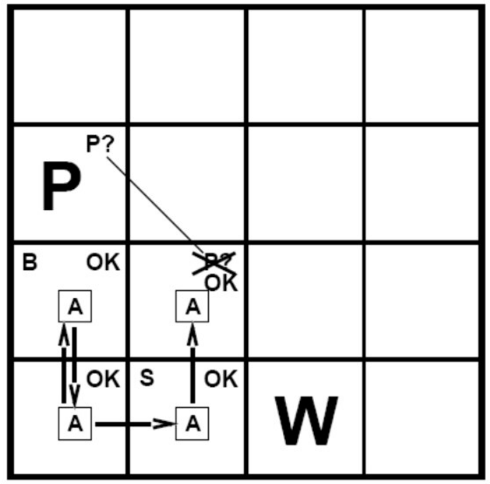
A = Πράκτορας
B = Αύρα
OK = Ασφαλές τετράγωνο
P = Παγίδα
S = Οσμή
G = Λάμψη



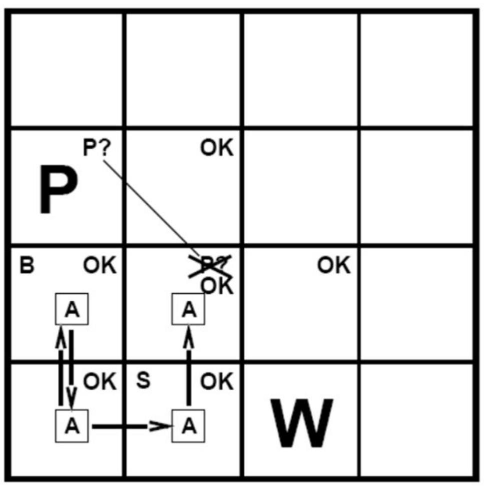


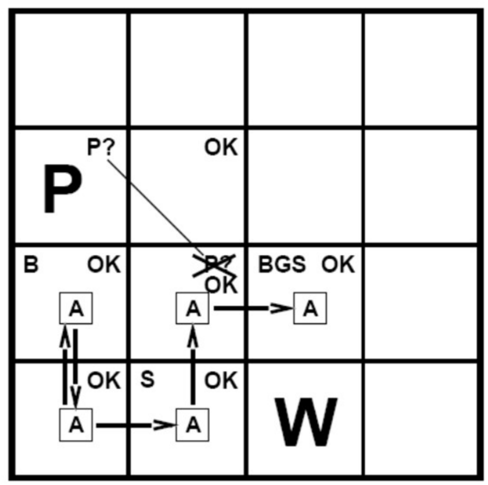
Εξερεύνηση στον κόσμο του Wumpus





A = Πράκτορας
B = Αύρα
OK = Ασφαλές τετράγωνο
P = Παγίδα
S = Οσμή
G = Λάμψη





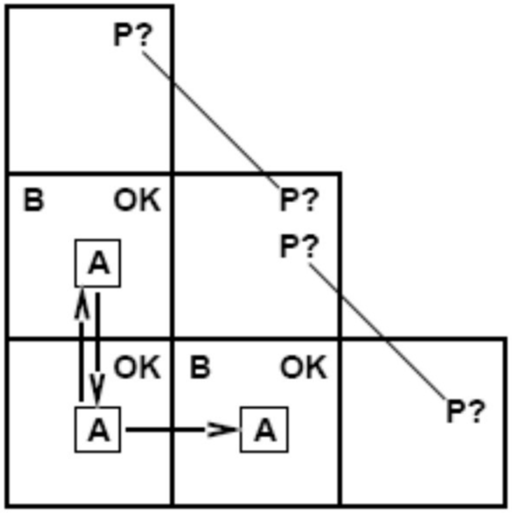
Εξερεύνηση στον κόσμο του Wumpus

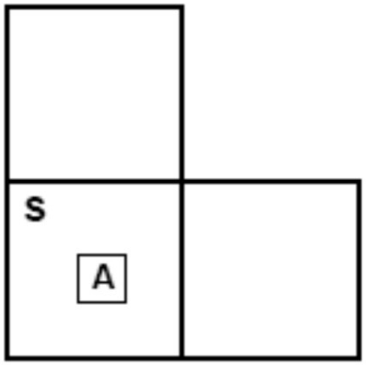
- Αν έχει αύρα στα [1,2] και [2,1] => δεν υπάρχουν ασφαλείς κινήσεις
  - αν υποθέσουμε ότι οι παγίδες είναι κατανεμημένες ομοιόμορφα τότε το [2,2] έχει παγίδα με πιθανότητα 0.86 (αντί για 0.31)
- Αν έχει δυσοσμία στο [1,1] => δεν μπορεί ο πράκτορας να κινηθεί
  - μπορεί να χρησιμοποιήσει την εξής στρατηγική:

    Ρίξε βέλος ευθεία

    Αν το Wumpus ήταν εκεί => νεκρό => ασφαλές

    Αν το Wumpus δεν ήταν εκεί => ασφαλές





Γενικά για Λογική

- Οι **λογικές** είναι επίσημες γλώσσες για την αναπαράσταση πληροφορίας
  - έτσι ώστε να μπορούν να εξαχθούν συμπεράσματα
- Η **σύνταξη** καθορίζει τη μορφή των προτάσεων της γλώσσας προσδιορίζοντας ποιες προτάσεις είναι καλά διατυπωμένες
- Η **σημασιολογία** καθορίζει τη σημασία των προτάσεων;
  - κατά πόσο οι προτάσεις ισχύουν (είναι αληθείς) σε κάθε **δυνατό κόσμο**
- <u>Για παράδειγμα η γλώσσα της αριθμητικής</u>
  - $x + 2 \geq y$ είναι μια καλά διατυπωμένη πρόταση; ενώ η $x2 + y >$ δεν είναι καλά διατυπωμένη πρόταση
  - $x + 2 \geq y$ είναι αληθές αν και μόνο αν το νούμερο $x + 2$ δεν είναι μικρότερο από το νούμερο y
  - $x + 2 \geq y$ είναι αληθές σε έναν κόσμο όπου $x=7$; $y =1$
  - $x + 2 \geq y$ ψευδές σε έναν κόσμο όπου $x=0$; $y =6$

Λογική

- Μια γλώσσα αναπαράστασης γνώσης (KR language) ορίζεται από την **σύνταξη** (syntax) και τη **σημασιολογία** (semantics) της
  - Η **σύνταξη** αποτελείται από το σύνολο των συμβόλων που χρησιμοποιεί η γλώσσα και των κανόνων σύμφωνα με του οποίους τα σύμβολα μπορούν να συνδυαστούν
  - Η **σημασιολογία** καθορίζει μια αντιστοιχία μεταξύ συμβόλων, συνδυασμών συμβόλων, προτάσεων της γλώσσας και εννοιών του κόσμου στις οποίες αναφέρονται

- Μια πρόταση μιας KR γλώσσας δε σημαίνει τίποτα από μόνη της
  - Η σημασιολογία (δηλ. το νόημα) της πρότασης πρέπει να προσδιοριστεί από συγγραφέα της γλώσσας μέσω μιας **ερμηνείας** (interpretation)

Μοντέλα

- Συνήθως στη λογική χρησιμοποιείται ο όρος **μοντέλο** αντί για "δυνατός κόσμος"
  - το *m* είναι μοντέλο της *α* θα σημαίνει ότι η πρόταση *α* είναι αληθής στο μοντέλο *m*
  - *M(α)* είναι το σύνολο όλων των μοντέλων της *α*
- Ένα μοντέλο είναι μια μαθηματική αφαίρεση ενός πραγματικού περιβάλλοντος
  - Π.χ. αν *x* και *y* είναι οι αριθμοί των ανδρών και γυναικών που παίζουν σε ένα παιχνίδι μπριτζ τότε οι πρόταση *x*+y=4 είναι αληθής όταν 4 συνολικά άτομα παίζουν
  - όλα τα δυνατά μοντέλα είναι όλες οι δυνατές αναθέσεις τιμών στις μεταβλητές *x* και *y* (μπορεί να είναι άπειρα σε πλήθος)
- Μια πρόταση λέγεται **αληθής** κάτω ένα συγκεκριμένο μοντέλο αν οι καταστάσεις του πραγματικού κόσμου τις οποίες αναπαριστά ισχύουν

Μοντέλα

Η πρόταση "*Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών εδρεύει στην Κοζάνη*" <u>είναι αληθής</u> σύμφωνα με το μοντέλο του κόσμου όπου:

το «*Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών*» αφορά το συγκεκριμένο τμήμα του Παν. Δυτ. Μακεδονίας

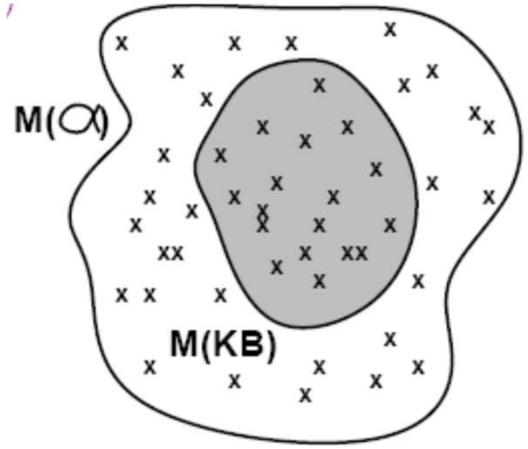
- η «*Κοζάνη*» αφορά την συγκεκριμένη πόλη της Δυτ. Μακεδονίας

- Η ίδια πρόταση <u>δεν είναι αληθής</u> κάτω από την ερμηνεία όπου
  - η «*Κοζάνη*» αφορά το ομώνυμο ζαχαροπλαστείο της Θεσ/νίκης

Λογική Κάλυψη (Entailment)

- Η **λογική κάλυψη** (entailment) απλά σημαίνει ότι μια πρόταση εξάγεται, προκύπτει λογικά, από κάποιες άλλες
- Επίσημα λέμε ότι μια πρόταση *α* **λογικά καλύπτεται** (is entailed) από τις προτάσεις μιας βάσης γνώσης (*ΒΓ*) όταν οποτεδήποτε οι προτάσεις της *ΒΓ* είναι αληθείς τότε και η *α* είναι αληθής
  - σε όλα τα μοντέλα που είναι αληθής η *ΒΓ* είναι αληθής και η *α*
  - Η λογική κάλυψη συμβολίζεται συνήθως ως *KB* |= *α*

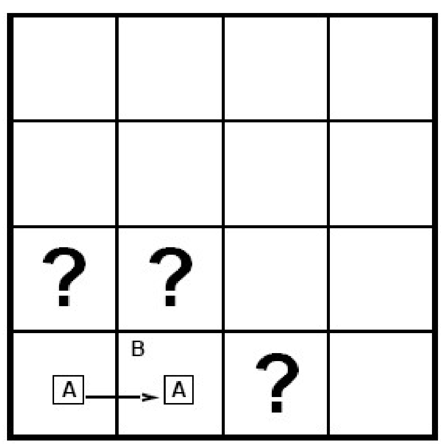
KB |= *α* αν και μόνο αν *M(KB)* ⊆ *M(A)*

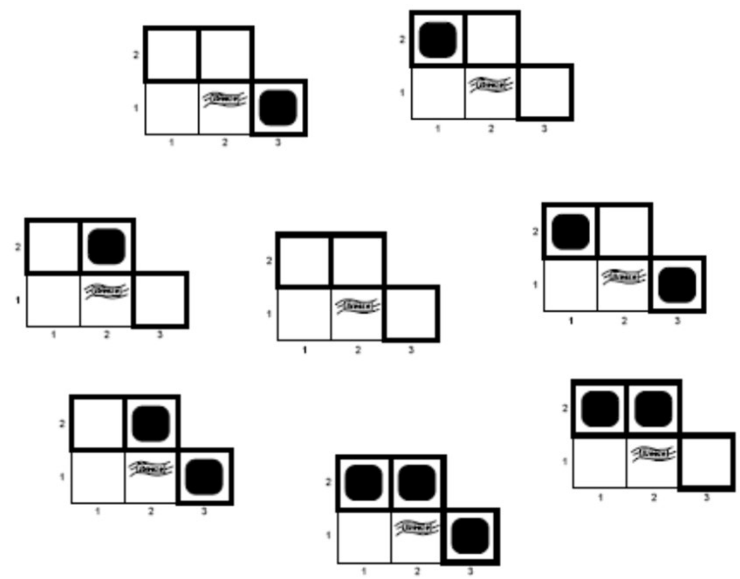


- "*Η Ρεάλ Μαδρίτης κέρδισε* και *η Μπαρτσελόνα κέρδισε*" |= "*Κέρδισε η Ρεάλ Μαδρίτης ή η Μπαρτσελόνα*"
- (x + y = 4) |= (4 = x + y)

Entailment στον κόσμο του Wumpus

- Η κατάσταση αφού ο πράκτορας δε βρει τίποτα στο [1,1], μετακινηθεί δεξιά, και βρει αύρα στο [2,1]
  - ποια είναι τα μοντέλα των ? τετραγώνων σε ό,τι αφορά παγίδες;
    - 3 δυαδικές επιλογές -> 8 πιθανά μοντέλα

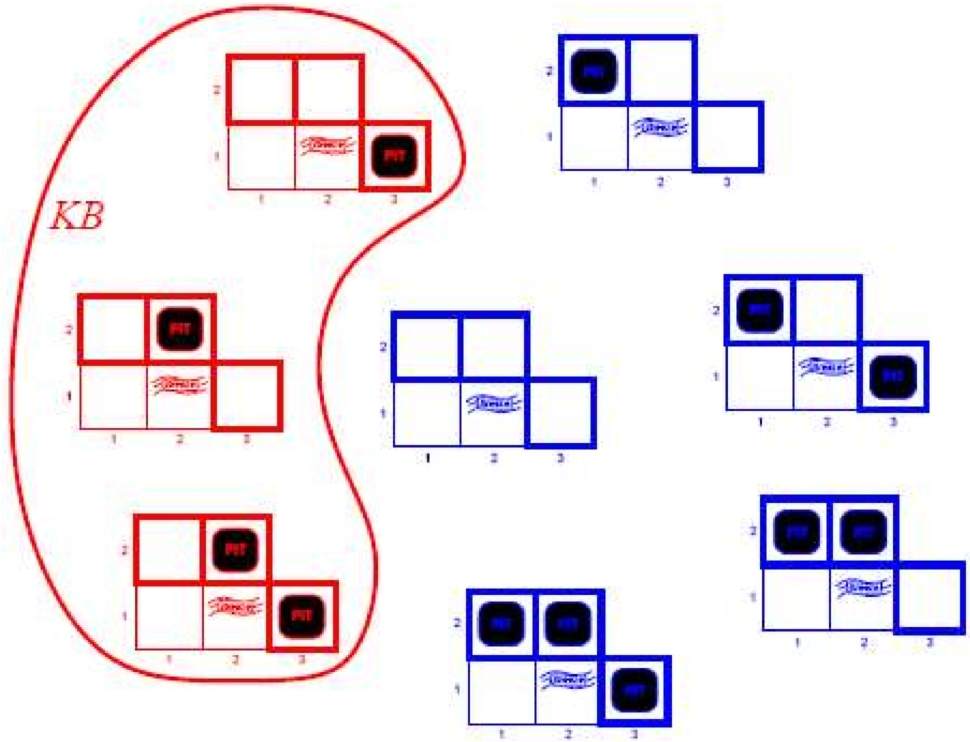




Entailment στον κόσμο του Wumpus

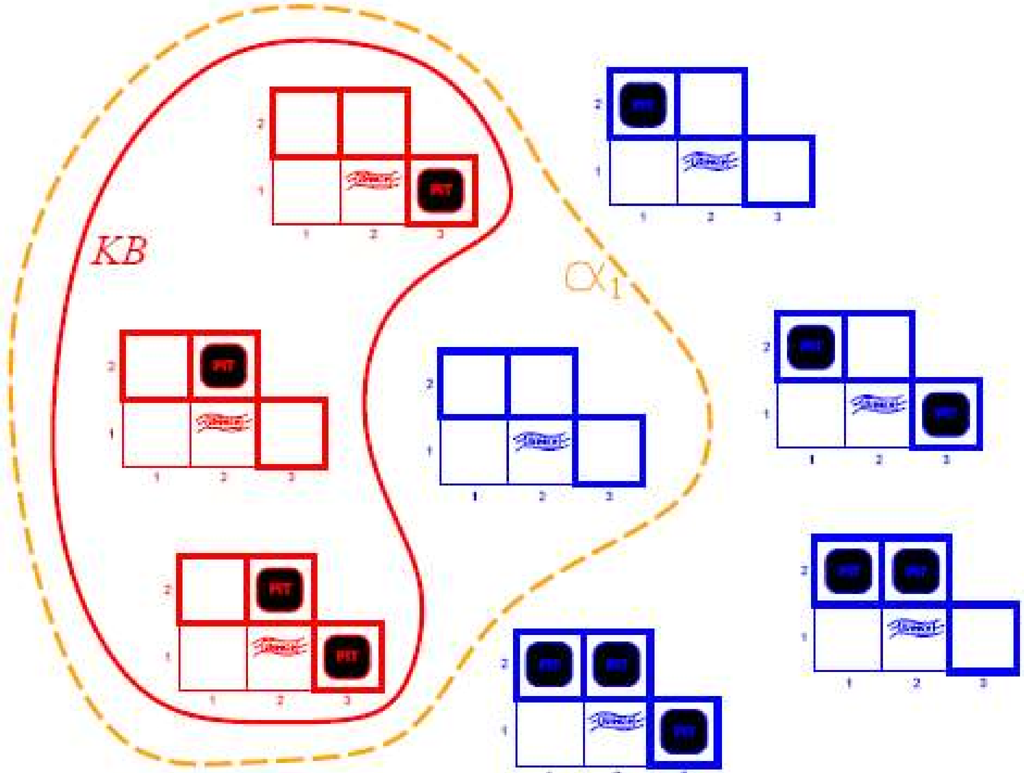
- Η Βάση Γνώσης αποτελείται από:
  - τους κανόνες του κόσμου του Wumpus
  - τα δεδομένα που ισχύουν σύμφωνα με τις παρατηρήσεις του πράκτορα

- <u>Υπάρχουν 3 μοντέλα για τα οποία η ΒΓ είναι αληθής</u>



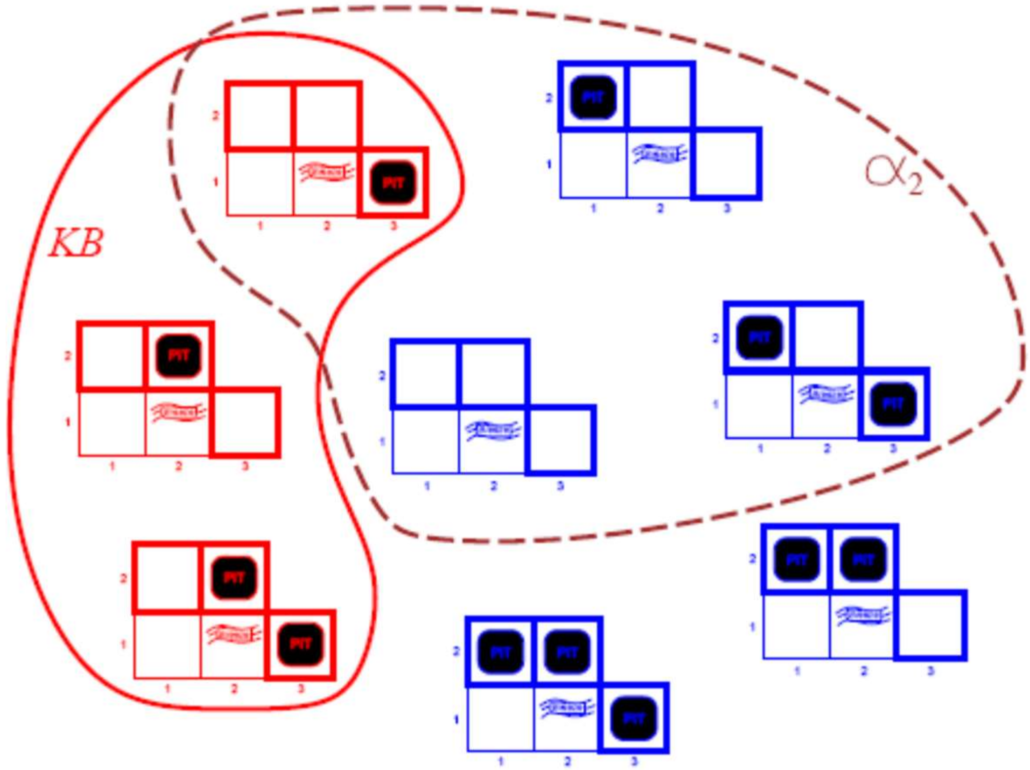
Entailment στον κόσμο του Wumpus

- Η Βάση Γνώσης αποτελείται από:
  - τους κανόνες του κόσμου του Wumpus
  - τα δεδομένα που ισχύουν σύμφωνα με τις παρατηρήσεις του πράκτορα
- Η πρόταση α1: "*το [1,2] είναι ασφαλές*" είναι entailed από τη ΒΓ
  - σε κάθε μοντέλο όπου η α1 είναι αληθής, η ΒΓ είναι αληθής



Entailment στον κόσμο του Wumpus

- Η πρόταση α2: “δεν υπάρχει παγίδα στο *[2,2]*” <u>δεν είναι entailed</u> (|≠) από τη ΒΓ
  - σε μερικά μοντέλα η ΒΓ είναι αληθής, η α2 είναι ψευδής
  - άρα ο πράκτορας δεν μπορεί να συμπεράνει ότι δεν υπάρχει παγίδα στο [2,2].
    - Χωρίς αυτό να σημαίνει ότι σίγουρα υπάρχει



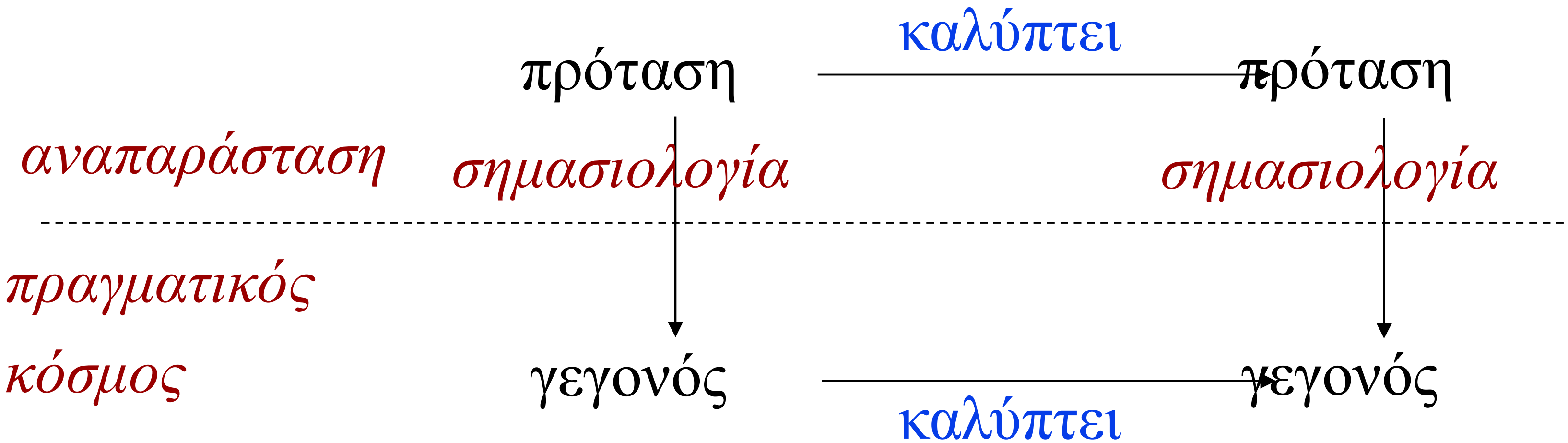
Εξαγωγή Συμπερασμάτων & Αποδείξεις

- **Ο Λογικός Συμπερασμός** (logical inference) είναι η διαδικασία της μηχανικής κατασκευής προτάσεων που εξάγονται από μια βάση γνώσης
  - Μια μορφή λογικού συμπερασμού (**έλεγχος μοντέλων**) είναι η διαδικασία απαρίθμησης όλων των μοντέλων για να ελεγχθεί αν μια πρόταση *α* είναι αληθής σε όλα τα μοντέλα όπου μια *ΒΓ* είναι αληθής
  - Αν μια πρόταση *α* εξάγεται από μια *ΒΓ* χρησιμοποιώντας έναν μηχανισμό εξαγωγής συμπερασμάτων i, γράφουμε $ΒΓ \vdash_i α$
  - Ένας μηχανισμός εξαγωγής συμπερασμάτων είναι **ορθός** (sound) αν κατασκευάζει μόνο προτάσεις που εξάγονται από τη *ΒΓ*
    - *ο έλεγχος μοντέλων είναι ορθός?*
  - Ένας μηχανισμός εξαγωγής συμπερασμάτων είναι **πλήρης** (complete) αν κατασκευάζει όλες τις προτάσεις που εξάγονται από τη *ΒΓ*

Εξαγωγή Συμπερασμάτων & Αποδείξεις

- Τα βήματα που απαιτούνται για να δημιουργηθεί μια πρόταση *α* από ένα σύνολο προτάσεων *ΒΓ* ονομάζεται **απόδειξη** (proof)
  - **Θεωρία απόδειξης** (proof theory) είναι ένα σύνολο από κανόνες για τη δημιουργία εξαγόμενων προτάσεων από ένα σύνολο προτάσεων

- Αν και μια διαδικασία συμπερασμού επενεργεί στη σύνταξη (στις προτάσεις της γλώσσας λογικής), υπάρχει σαφής σχέση με τον πραγματικό κόσμο



Γλώσσες Αναπαράστασης Γνώσης

- Θα κάνουμε μια σύντομη επισκόπηση μιας βασικής γλώσσας αναπαράστασης γνώσης:
  - **Προτασιακή Λογική** (Propositional Logic)
- Άλλες γλώσσες λογικής:
  - **Λογική Πρώτης Τάξης** ή **Κατηγορηματική Λογική** (First Order Logic or Predicate Logic)

- Γενικά μια **λογική** είναι ένα σύστημα που αποτελείται από:
  - **Σύνταξη**
  - **Σημασιολογία**
  - **Θεωρία Απόδειξης**

- Γιατί δε χρησιμοποιούμε φυσική γλώσσα ή γλώσσες προγραμματισμού για αναπαράσταση γνώσης ?

Προτασιακή Λογική - Σύνταξη

- Ατομικές προτάσεις
  - **Προτασιακά σύμβολα:** P, Q, R, $W_{1,3}$, $\Gamma_{3,1}$, *Αληθές*, *Ψευδές*
    - Π.χ. το σύμβολο W1,3 αντιπροσωπεύει την ατομική πρόταση «το Wumpus είναι στο τετράγωνο [1,3]
- Λογικά συνδετικά
  - **Άρνηση,** $\neg W_{1,3}$
    - Λεκτικά (literals), Θετικό λεκτικό: $W_{1,3}$, Αρνητικό λεκτικό:, $\neg W_{1,3}$
  - **Σύζευξη,** $W_{1,3} \land \Gamma_{3,1}$
    - Συζευκτέοι
  - **Διάζευξη,** $(W_{1,3} \land \Gamma_{3,1}) \lor W_{2,2}$
    - Διαζευκτέοι
  - **Συνεπαγωγή,** $(W_{1,3} \land \Gamma_{3,1}) \Rightarrow \neg W_{2,2}$
    - προϋπόθεση ή προηγούμενο, συμπέρασμα ή επακόλουθο
    - κανόνες, προτάσεις εάν-τότε
  - **Ισοδυναμία,** $W_{1,3} \Leftrightarrow \neg W_{2,2}$

Προτασιακή Λογική - Σύνταξη

- Η παρακάτω BNF γραμματική ορίζει τις **καλά σχηματισμένες προτάσεις** (well-formed sentences) της ΠΛ

$$
\begin{aligned}
&\text{Πρόταση} \rightarrow \text{ΑτομικήΠρόταση} \mid \text{ΠερίπλοκηΠρόταση} \\
&\text{ΑτομικήΠρόταση} \rightarrow P_1 \mid P_2 \mid \ldots \\
&\text{ΠερίπλοκηΠρόταση} \rightarrow (\text{Πρόταση}) \mid \neg\, \text{Πρόταση} \mid \\
&\qquad\qquad \text{Πρόταση ΔυαδικόΣυνδετικό Πρόταση} \\
&\text{ΔυαδικόΣυνδετικό} \rightarrow \wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow
\end{aligned}
$$

- Η γραμματική είναι πολύ αυστηρή με τις παρενθέσεις
  - π.χ. πρέπει να γράφουμε $((A \wedge B) \Rightarrow \Gamma)$ κι όχι $A \wedge B \Rightarrow \Gamma$
- Η προτεραιότητα των συνδετικών βοηθάει στην αναγνωσιμότητα
  - Προτεραιότητα: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
    - Π.χ. $\neg P \vee Q \wedge R \Rightarrow S$ ισοδυναμεί με $((\neg P) \vee (Q \wedge R)) \Rightarrow S$

Προτασιακή Λογική - Σημασιολογία

- Ένα **προτασιακό σύμβολο** μπορεί να συμβολίζει οτιδήποτε θέλουμε και μπορεί να είναι αληθές ή ψευδές.
  - Δηλαδή η ερμηνεία του μπορεί να είναι οποιοδήποτε γεγονός ή έννοια του πραγματικού κόσμου
  - Το γεγονός αυτό θα είναι είτε αληθές είτε ψευδές στον πραγματικό κόσμο
  - Τι ακριβώς είναι η ερμηνεία ?
- Ας θεωρήσουμε ένα σύνολο P προτασιακών συμβόλων. Μια **ερμηνεία** (interpretation) του P είναι μια αντιστοίχιση

  $$I : P \rightarrow \{\text{true, false}\}$$

  Δηλαδή καθορίζει ένα μοντέλο για τα σύμβολα
  - για παράδειγμα *W1,1 = false, W2,1 = false, W1,2 = true*
- Η έννοια της ερμηνείας μπορεί να επεκταθεί σε οποιαδήποτε καλά σχηματισμένη πρόταση χρησιμοποιώντας την ερμηνεία των λογικών συνδετικών

Προτασιακή Λογική - Σημασιολογία

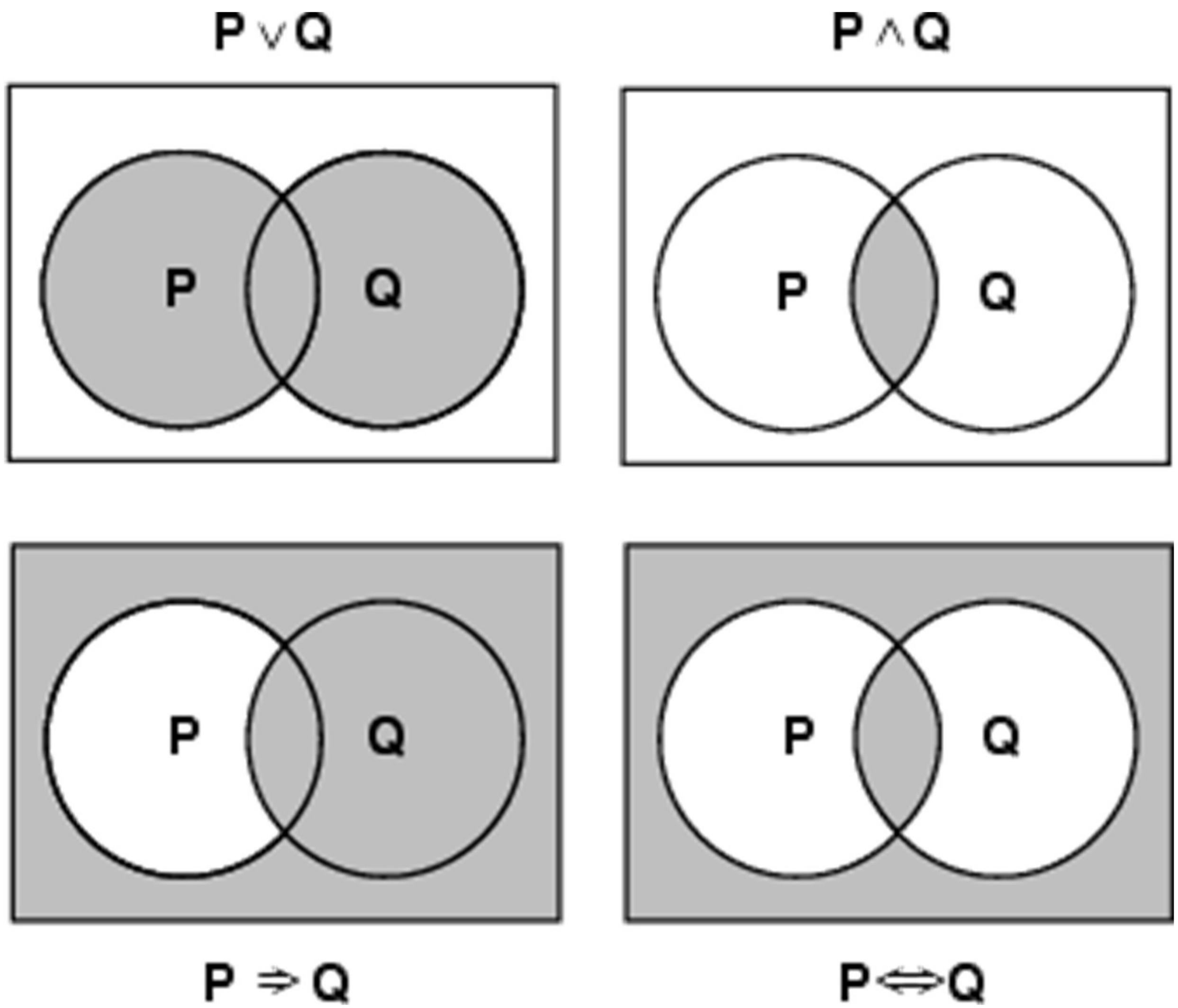
- Η σημασιολογία της ΠΛ πρέπει να καθορίζει πως υπολογίζεται η τιμή αληθείας οποιασδήποτε πρότασης με δεδομένο ένα μοντέλο
  - Εφόσον όλες οι προτάσεις κατασκευάζονται από ατομικές προτάσεις και τα συνδετικά, πρέπει να καθορίσουμε:
    1. πως υπολογίζεται η αλήθεια των ατομικών προτάσεων
    2. πως υπολογίζεται η αλήθεια των προτάσεων που σχηματίζονται με καθένα από τα συνδετικά
  - **Το πρώτο είναι απλό**
    - Η *true* είναι αληθής σε κάθε μοντέλο και η *false* είναι ψευδής σε κάθε μοντέλο
    - Η τιμή αλήθειας κάθε άλλου προτασιακού συμβόλου καθορίζεται άμεσα στο μοντέλο (π.χ. $W1,1 = false$, $W2,1 = false$, $W1,2 = true$)
  - **Για το δεύτερο έχουμε κανόνες**
    - Π.χ. Για οποιαδήποτε πρόταση *s* και μοντέλο *m*, η πρόταση $\neg s$ είναι αληθής στο m εάν και μόνο η *s* είναι ψευδής στο *m*

Σημασιολογία – Πίνακες Αληθείας

- Οι κανόνες για το κάθε συνδετικό μπορούν να συνοψιστούν σε έναν **πίνακα αληθείας** (truth table)
  - με χρήση των πινάκων μπορεί να υπολογιστεί η τιμή αληθείας κάθε πρότασης αναδρομικά

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
ψευδές	*ψευδές*	*αληθές*	*ψευδές*	*ψευδές*	*αληθές*	*αληθές*
ψευδές	*αληθές*	*αληθές*	*ψευδές*	*αληθές*	*αληθές*	*ψευδές*
αληθές	*ψευδές*	*ψευδές*	*ψευδές*	*αληθές*	*ψευδές*	*ψευδές*

Σημασιολογία – Πίνακες Αληθείας



Σημασιολογία – Πίνακες Αληθείας

- Οι πίνακες αληθείας είναι σύμφωνοι με την ανθρώπινη διαίσθηση
  - προσοχή στο $P \vee Q$ (είναι αληθές όταν P και Q είναι αληθή)

εκτός από την περίπτωση του $P \Rightarrow Q$

  - P *συνεπάγεται* Q (αν P *τότε* Q)
    - Στην ΠΛ δεν χρειάζεται να υπάρχει σχέση αιτιότητας ή συνάφειας μεταξύ P και Q. Η πρόταση *«βρέχει στην Κοζάνη => η Κρήτη είναι νησί»* είναι απόλυτα σωστή στην ΠΛ
    - οποιαδήποτε συνεπαγωγή είναι αληθής όταν η προϋπόθεση της είναι ψευδής. Π.χ. «η Κοζάνη είναι νησί => θα χιονίσει τα Χριστούγεννα» είναι αληθής πρόταση άσχετα με το αν θα χιονίσει τα Χριστούγεννα ή όχι.
    - $P \Rightarrow Q$ *στην ΠΛ μας λέει «Αν ισχύει το P ισχυρίζομαι ότι είναι αληθές το Q, αλλιώς δεν ισχυρίζομαι τίποτα»*
      - Αυτό είναι ψευδές μόνο αν το P είναι αληθές και το Q ψευδές

Σημασιολογία – Πίνακες Αληθείας

- Ο πίνακας αληθείας για την ισοδυναμία δείχνει ότι το $P \Leftrightarrow Q$ είναι αληθές όταν είναι αληθές το $P \Rightarrow Q$ και το $Q \Rightarrow P$
- Ας σκεφτούμε τους κανόνες του κόσμου του Wumpus. Πως θα γράψουμε τον κανόνα που μας λέει ότι ένα τετράγωνο έχει αύρα αν ένα γειτονικό τετράγωνο έχει παγίδα;

  $B1,1 \Leftrightarrow (P1,2 \lor P2,1)$

  ή

  $B1,1 \Rightarrow (P1,2 \lor P2,1)$

  - Η συνεπαγωγή απαιτεί να υπάρχει παγίδα αν υπάρχει αύρα αλλά δεν αποκλείει να υπάρχει παγίδα όταν δεν υπάρχει αύρα
    - αυτό το κάνει η ισοδυναμία

Μια απλή βάση γνώσης

- Θα ασχοληθούμε μόνο με τις παγίδες (γούβες):
  - R_1: $\neg \Gamma_{1,1}$
  - R_2: $A_{1,1} \Leftrightarrow (\Gamma_{1,2} \vee \Gamma_{2,1})$.
  - R_3: $A_{2,1} \Leftrightarrow (\Gamma_{1,1} \vee \Gamma_{2,2} \vee \Gamma_{3,1})$.
  - R_4: $\neg A_{1,1}$
  - R_5: $A_{2,1}$
  - R_6: $\neg \Gamma_{2,1}$

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 Γ?	3,2	4,2
1,1 E OK	2,1 Π A OK	3,1 Γ?	4,1

- Βάση γνώσης: Σύζευξη προτάσεων

  $KB \equiv R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 \wedge R_6$

Συμπερασμός με απαρίθμηση

- Θέλουμε να απαντάμε σε ερωτήσεις της μορφής: $KB \models \alpha$?
- 7 μεταβλητές: $A_{1,1}$, $A_{2,1}$, $\Gamma_{1,1}$, $\Gamma_{1,2}$, $\Gamma_{2,1}$, $\Gamma_{2,2}$ και $\Gamma_{3,1}$
- 2^7=128 δυνατά μοντέλα
- Η *KB* είναι αληθής σε 3 από αυτά.
  - $KB \models \neg\Gamma_{1,2}$
  - $KB \not\models \Gamma_{2,2}$
  - $KB \not\models \neg\Gamma_{2,2}$
- Χρονική πολυπλοκότητα: $O(2^n)$
- Χωρική πολυπλοκότητα: $O(n)$
  - όπου n το πλήθος των προτασιακών συμβόλων
- *Κάθε γνωστός πλήρης αλγόριθμος συμπερασμού για την προτασιακή λογική έχει μια πολυπλοκότητα χειρότερης περίπτωσης που είναι εκθετική ως προς το μέγεθος της εισόδου.*

Λογική ισοδυναμία

Δύο προτάσεις είναι λογικά ισοδύναμες αν είναι αληθείς στο ίδιο σύνολο μοντέλων

- $(\alpha \land \beta) \equiv (\beta \land \alpha)$ — αντιμεταθετικότητα του $\land$
- $(\alpha \lor \beta) \equiv (\beta \lor \alpha)$ — αντιμεταθετικότητα του $\lor$
- $((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma))$ — προσεταιριστικότητα του $\land$
- $((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma))$ — προσεταιριστικότητα του $\lor$
- $\neg(\neg\alpha) \equiv \alpha$ — απαλοιφή διπλής άρνησης
- $(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ — αντιθετοαντιστροφή
- $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \lor \beta)$ — απαλοιφή συνεπαγωγής
- $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$ — απαλοιφή αμφίδρομης υποθετικής πρότασης
- $\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$ — νόμος De Morgan
- $\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta)$ — νόμος De Morgan
- $(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (a \land \gamma))$ — επιμεριστικότητα του $\land$ ως προς το $\lor$
- $(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (a \lor \gamma))$ — επιμεριστικότητα του $\lor$ ως προς το $\land$

Έγκυρες και Ικανοποιήσιμες προτάσεις

- **Έγκυρες προτάσεις**: Είναι αληθείς σε όλα τα πιθανά μοντέλα
  - π.χ. $P \lor \neg P$
- **Ικανοποιήσιμες προτάσεις**: Είναι αληθείς σε τουλάχιστον ένα μοντέλο.
  - Η πρόταση α είναι αληθής στο μοντέλο m. Το m ικανοποιεί την α. Το m είναι ένα μοντέλο της α
  - π.χ. P, $P \lor Q$, $(P \land R) \lor Q$
- **μη ικανοποιήσιμες προτάσεις**: δεν είναι αληθείς σε κανένα μοντέλο
  - π.χ. $P \land \neg P$
- Η α είναι έγκυρη εάν και μόνο αν η $\neg\alpha$ δεν είναι ικανοποιήσιμη.
- Απαγωγή σε άτοπο:
  - *$\alpha \vDash \beta$ εάν και μόνο εάν η πρόταση $(\alpha \land \neg\beta)$ είναι μη ικανοποιήσιμη.*

Πολυπλοκότητα στην Προτασιακή Λογική

- Θεώρημα: Το πρόβλημα του καθορισμού αν μια πρόταση της ΠΛ είναι ικανοποιήσιμη είναι **NP-complete** (Cook, 1971)
  - Το πρόβλημα του καθορισμού αν μια πρόταση του ΠΛ είναι έγκυρη είναι **co-NP-complete**
  - Είναι πολύ απίθανο να βρούμε έναν πολυωνυμικό αλγόριθμο για αυτά τα προβλήματα

- Μια πρόταση της ΠΛ καλείται **πρόταση Horn** αν είναι της μορφής

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$$

ή ισοδύναμα $\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n \vee Q$

- Θεώρημα: Αν η φ είναι μια σύζευξη προτάσεων Horn τότε το αν η φ είναι ικανοποιήσιμη μπορεί να βρεθεί σε πολυωνυμικό χρόνο

Κανόνες Εξαγωγής Συμπερασμάτων στην ΠΛ

- Ένας κανόνας εξαγωγής συμπερασμάτων είναι ένας κανόνας της μορφής

  $$\alpha_1, \alpha_2, \ldots, \alpha_n \models \beta$$

  όπου οι $\alpha_1, \alpha_2, \ldots, \alpha_n$ είναι προτάσεις που ονομάζονται **συνθήκες** (conditions) και η β είναι μια πρόταση που ονομάζεται **συμπέρασμα** (conclusion)

- Όποτε έχουμε ένα σύνολο προτάσεων που ταιριάζουν με τις συνθήκες ενός κανόνα τότε μπορούμε να εξάγουμε την πρόταση που είναι το συμπέρασμα του κανόνα

Κανόνες Εξαγωγής Συμπερασμάτων στην ΠΛ

- **Modus Ponens (τρόπος του «θέτειν»):** $(\alpha \Rightarrow \beta) \wedge \alpha \models \beta$
- **And-Elimination (απαλοιφή του «και»):** $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n \models \alpha_i$
- **And-Introduction (εισαγωγή του «και»):** $\alpha_1, \alpha_2, \ldots, \alpha_n \models \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$
- **Or-Introduction (εισαγωγή του «ή»):** $\alpha_i \models \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n$
- **Double-Negation Elimination (απαλοιφή διπλής άρνησης) :** $\neg\neg\alpha \models \alpha$
- **Unit Resolution (μοναδιαία ανάλυση):** $(\alpha \vee \beta) \wedge \neg\beta \models \alpha$
- **Resolution (ανάλυση):** $(\alpha \vee \beta) \wedge (\neg\beta \vee \gamma) \models (\alpha \vee \gamma)$

Πρότυπα συλλογιστικής στην προτασιακή λογική

Αποδείξεις

Ένας κανόνας εξαγωγής συμπερασμάτων είναι ένας κανόνας της μορφής

$\alpha_1, \alpha_2, \ldots, \alpha_n \models \beta$

όπου οι $\alpha_1, \alpha_2, \ldots, \alpha_n$ είναι προτάσεις που ονομάζονται **συνθήκες** (conditions) και η β είναι μια πρόταση που ονομάζεται **συμπέρασμα** (conclusion)

- Μια **απόδειξη** είναι μια σειρά προτάσεων, όπου κάθε πρόταση είναι είτε μια συνθήκη είτε μια πρόταση που εξάγεται από προηγούμενες προτάσεις στην απόδειξη χρησιμοποιώντας έναν κανόνα εξαγωγής συμπερασμάτων

Παράδειγμα Απόδειξης

Έστω οι παρακάτω προτάσεις (συνθήκες) σε μια Βάση Γνώσης:

(P ^ Q) => R

“Αν έχει ζέστη και υγρασία, τότε βρέχει”

Q => P

“Αν έχει υγρασία, τότε έχει ζέστη”

Q

“Έχει υγρασία”

Εξάγοντας Συμπεράσματα

- Μια **απόδειξη** είναι μια σειρά προτάσεων, όπου κάθε πρόταση είναι είτε μια συνθήκη είτε μια πρόταση που εξάγεται από προηγούμενες προτάσεις στην απόδειξη χρησιμοποιώντας έναν κανόνα εξαγωγής συμπερασμάτων

- Παράδειγμα:
  1. Q **Συνθήκη** "Έχει υγρασία"
  2. Q => P **Συνθήκη** "Αν έχει υγρασία, τότε έχει ζέστη"
  3. P **Modus Ponens(1,2)** "Έχει ζέστη"
  4. (P ^ Q) => R **Συνθήκη** "Αν έχει ζέστη και υγρασία , τότε βρέχει"
  5. P ^ Q **And Introduction(1,3)** "Έχει ζέστη και υγρασία "
  6. R **Modus Ponens(4,5)** "Βρέχει"

Εξάγοντας Συμπεράσματα

Έχοντας ορίσει τον μηχανισμό απόδειξης, μπορούμε να τον χρησιμοποιήσουμε για να απαντήσουμε σε ερωτήματα (queries)

“*Έχει ζέστη* ?”

“*Βρέχει* ?”

Φυσικά χρειαζόμαστε μεθόδους/αλγόριθμους που χρησιμοποιούν τους κανόνες εξαγωγής συμπερασμάτων με δομημένο / έξυπνο τρόπο

resolution

forward / backward chaining

Σε περιπτώσεις όπου μας ενδιαφέρει να μάθουμε αν ένα σύνολο προτάσεων είναι ικανοιποιήσιμο ή όχι (δηλ. αν έχει μοντέλο) υπάρχουν πολύ πιο αποδοτικοί αλγόριθμοι

SAT algorithms!!!

Μέθοδοι Απόδειξης

- Οι μέθοδοι απόδειξης χωρίζονται σε δύο γενικές κατηγορίες:
  - **Εφαρμογή κανόνων συμπερασμού**
    - Ορθός τρόπος παραγωγής νέων προτάσεων από υπάρχουσες
    - Απόδειξη = μια ακολουθία εφαρμογής κανόνων συμπερασμού
    - Μπορούμε να χρησιμοποιήσουμε τους κανόνες συμπερασμού ως τις πιθανές ενέργειες σε έναν γενικό αλγόριθμο αναζήτησης
  - **Επαλήθευση Μοντέλων**
    - Απαρίθμηση πίνακα αληθείας (πάντα με εκθετικό κόστος)
    - βελτιωμένη αναζήτηση υπαναχώρησης (Davis-Putnam-Logemann-Loveland αλγόριθμος)
    - τοπική αναζήτηση στον χώρο μοντέλων (ορθή αλλά όχι πλήρης)
      - π.χ., hill-climbing αλγόριθμοι

Κανόνες συμπερασμού

- **Modus ponens** («τρόπος του θέτειν»)

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

- **Απαλοιφή του ΚΑΙ**

$$\frac{\alpha \wedge \beta}{\alpha}$$

- **Όλες οι λογικές ισοδυναμίες**, π.χ.

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \qquad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Παράδειγμα (1/3)

- Θα αποδείξουμε το $\neg \Gamma_{1,2}$:
  - R_6: $(A_{1,1} \Rightarrow (\Gamma_{1,2} \lor \Gamma_{2,1})) \land ((\Gamma_{1,2} \lor \Gamma_{2,1}) \Rightarrow A_{1,1})$
  - R_7: $((\Gamma_{1,2} \lor \Gamma_{2,1}) \Rightarrow A_{1,1})$
  - R_8: $(\neg A_{1,1} \Rightarrow \neg(\Gamma_{1,2} \lor \Gamma_{2,1}))$
  - R_9: $\neg(\Gamma_{1,2} \lor \Gamma_{2,1})$
  - R_{10}: $\neg \Gamma_{1,2} \land \neg \Gamma_{2,1}$
  - R_{11}: $\neg \Gamma_{1,2}$

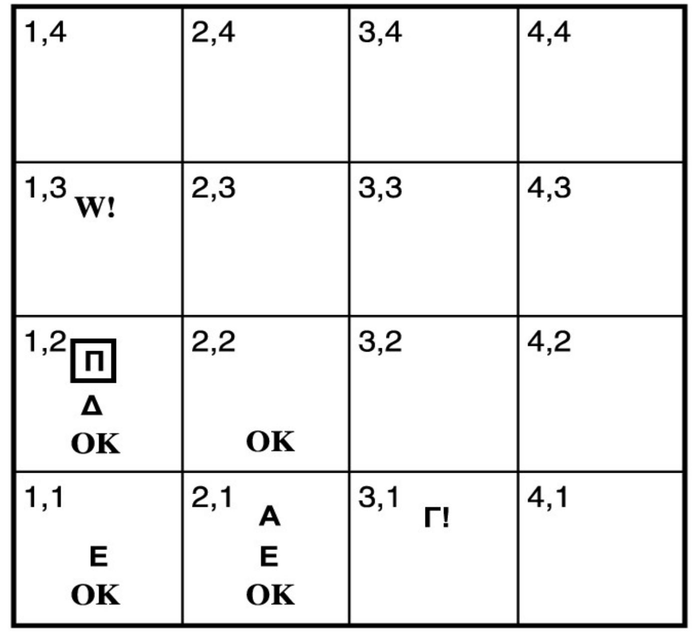
R_1: $\neg \Gamma_{1,1}$
R_2: $A_{1,1} \Leftrightarrow (\Gamma_{1,2} \lor \Gamma_{2,1})$
R_3: $A_{2,1} \Leftrightarrow (\Gamma_{1,1} \lor \Gamma_{2,2} \lor \Gamma_{3,1})$
R_4: $\neg A_{1,1}$
R_5: $A_{2,1}$

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2 **Γ?**	3,2	4,2
1,1 **E** **OK**	2,1 **Π** **A** **OK**	3,1 **Γ?**	4,1

Αποδείξεις

- Η παραπάνω συλλογιστική διαδικασία ονομάζεται **απόδειξη**
  - Η εύρεση αποδείξεων είναι ακριβώς σαν την εύρεση λύσεων σε προβλήματα αναζήτησης
    - Αρχική κατάσταση? Μετάβαση μεταξύ καταστάσεων (συνάρτηση διαδόχων)?
    - μπορούν να εφαρμοστούν όλοι οι γενικοί αλγόριθμοι αναζήτησης
- Διαδικασία αναζήτησης
  - **forward chaining**
  - **backward chaining**
  - είναι πολύ πιο αποδοτική από την απαρίθμηση μοντέλων στην πράξη
    - επειδή αγνοεί πολλές άσχετες ατομικές προτάσεις.
- **Μονοτονικότητα**
  - εάν $KB \vDash \alpha$ τότε $KB \wedge \beta \vDash \alpha$

Παράδειγμα (2/3)



- Από τις προτάσεις:
  - R_1: $\neg \Gamma_{1,1}$
  - R_2: $A_{1,1} \Leftrightarrow (\Gamma_{1,2} \vee \Gamma_{2,1})$
  - R_3: $A_{2,1} \Leftrightarrow (\Gamma_{1,1} \vee \Gamma_{2,2} \vee \Gamma_{3,1})$
  - R_4: $\neg A_{1,1}$
  - R_5: $A_{2,1}$
  - R_6: $(A_{1,1} \Rightarrow (\Gamma_{1,2} \vee \Gamma_{2,1})) \wedge ((\Gamma_{1,2} \vee \Gamma_{2,1}) \Rightarrow A_{1,1})$
  - R_7: $((\Gamma_{1,2} \vee \Gamma_{2,1}) \Rightarrow A_{1,1})$
  - R_8: $(\neg A_{1,1} \Rightarrow \neg(\Gamma_{1,2} \vee \Gamma_{2,1}))$
  - R_9: $\neg(\Gamma_{1,2} \vee \Gamma_{2,1})$
  - R_{10}: $\neg \Gamma_{1,2} \wedge \neg \Gamma_{2,1}$
- και πηγαίνοντας από το [2,1] στο [1,1] και μετά στο [1,2], όπου υπάρχει δυσοσμία αλλά όχι αύρα, προκύπτουν…

Παράδειγμα (3/3)

- (συνέχεια…)
  - R_{11}: $\neg A_{1,2}$
  - R_{12}: $A_{1,2} \Leftrightarrow (\Gamma_{1,1} \vee \Gamma_{2,2} \vee \Gamma_{1,3})$
- Εφαρμόζοντας αντιθετοαντιστροφή και modus ponens παίρνουμε:
  - R_{13}: $\neg \Gamma_{2,2}$
  - R_{14}: $\neg \Gamma_{1,3}$
- Από τις R_3 και R_5 παίρνουμε:
  - R_{15}: $\Gamma_{1,1} \vee \Gamma_{2,2} \vee \Gamma_{3,1}$
- Από την R_{15} και την R_{13} παίρνουμε:
  - R_{16}: $\Gamma_{1,1} \vee \Gamma_{3,1}$
- Τέλος από την R_{16} και την R_1 παίρνουμε:
  - R_{17}: $\Gamma_{3,1}$

Ανάλυση
(resolution)

- **Μοναδιαία ανάλυση** (unit resolution)

$$\frac{l_1 \vee \square \vee l_k, \qquad m}{l_1 \vee \square \vee l_{i-1} \vee l_{i+1} \vee \square \vee l_k} \qquad\qquad l_i = \neg m$$

- **Πλήρης ανάλυση**

$$\frac{l_1 \vee \square \vee l_k, \qquad m_1 \vee \square \vee m_n}{l_1 \vee \square \vee l_{i-1} \vee l_{i+1} \vee \square \vee l_k \vee m_1 \vee \square \vee m_{j-1} \vee m_{j+1} \vee \square \vee m_n} \qquad l_i = \neg m_j$$

- Διαζευκτική πρόταση, συμπληρωματικά λεκτικά
- Παράδειγμα: $\dfrac{\Gamma_{1,1} \vee \Gamma_{3,1}, \qquad \neg \Gamma_{1,1} \vee \neg \Gamma_{2,2}}{\Gamma_{3,1} \vee \neg \Gamma_{2,2}}$

Πληρότητα

- Οποιοσδήποτε πλήρης αλγόριθμος αναζήτησης που εφαρμόζει μόνο τον κανόνα της ανάλυσης μπορεί να συνάγει οποιοδήποτε συμπέρασμα που καλύπτεται λογικά από οποιαδήποτε βάση γνώσης της προτασιακής λογικής.

- Με δεδομένο το A δεν μπορεί να «αποδείξει» το $A \lor B$.

- Μπορεί όμως να απαντήσει εάν το $A \lor B$ είναι αληθές ή ψευδές.
  - Πληρότητα διάψευσης

Συζευκτική Κανονική Μορφή

Κάθε πρόταση στην προτασιακή λογική μπορεί να μετασχηματιστεί σε **Συζευκτική Κανονική Μορφή** (*Conjunctive Normal Form – CNF*)

δηλαδή μια σύζευξη διαζεύξεων

Αλγόριθμος

Απάλειψε το $\Rightarrow$ χρησιμοποιώντας τον κανόνα **“($p \Rightarrow q$) είναι ισοδύναμο με ($\neg p \vee q$)”**

Χρησιμοποίησε τους κανόνες του **De Morgan** ώστε η άρνηση ($\neg$) να εφαρμόζεται μόνο σε ατομικά προτασιακά σύμβολα

Κατάνειμε τα $\vee$ και $\wedge$ για να έχεις στο τέλος μια σύζευξη διαζεύξεων

Συζευκτική Κανονική Μορφή - Παράδειγμα

$$\neg(p \Rightarrow q) \vee (\mathrm{r} \Rightarrow p)$$

Απάλειψε τις συνεπαγωγές

$\neg(\neg p \vee q) \vee (\neg r \vee p)$

Εφάρμοσε κανόνες de Morgan

$(p \wedge \neg q) \vee (\neg r \vee p)$

Εφάρμοσε κανόνες συσχέτισης και κατανομής

$(p \vee \neg r \vee p) \wedge (\neg q \vee \neg \mathrm{r} \vee p)$

$(p \vee \neg r) \wedge (\neg q \vee \neg \mathrm{r} \vee p)$

Συζευκτική κανονική μορφή
(conjunctive normal form, CNF)

- Κάθε πρόταση της προτασιακής λογικής είναι λογικά ισοδύναμη με μια σύζευξη διαζεύξεων λεκτικών.
  - Ένα *λεκτικό* είναι μια προτασιακή μεταβλητή ή η άρνηση της

Αλγόριθμος

- Απάλειψε τις $\Rightarrow$ με τον κανόνα $(p \Rightarrow q)$ ισοδυναμεί με $(\neg p \vee q)$
- Χρησιμοποίησε τους κανόνες de Morgan's ώστε οι αρνήσεις να εφαρμόζονται σε ατομικά λεκτικά
- Επιμερισμός των $\vee$ και $\wedge$ για να πάρουμε σύζευξη διαζεύξεων

- Διαδικασία μετατροπής σε CNF (παράδειγμα για R_2):
  - R_2: $A_{1,1} \Leftrightarrow (\Gamma_{1,2} \vee \Gamma_{2,1})$
  - $(A_{1,1} \Rightarrow (\Gamma_{1,2} \vee \Gamma_{2,1})) \wedge ((\Gamma_{1,2} \vee \Gamma_{2,1}) \Rightarrow A_{1,1})$
  - $(\neg A_{1,1} \vee \Gamma_{1,2} \vee \Gamma_{2,1}) \wedge (\neg(\Gamma_{1,2} \vee \Gamma_{2,1}) \vee A_{1,1})$
  - $(\neg A_{1,1} \vee \Gamma_{1,2} \vee \Gamma_{2,1}) \wedge ((\neg \Gamma_{1,2} \wedge \neg \Gamma_{2,1}) \vee A_{1,1})$
  - $(\neg A_{1,1} \vee \Gamma_{1,2} \vee \Gamma_{2,1}) \wedge (\neg \Gamma_{1,2} \vee A_{1,1}) \wedge (\neg \Gamma_{2,1} \vee A_{1,1})$

Αλγόριθμος ανάλυσης

- Για να αποδείξουμε το $KB \vDash \alpha$, αποδεικνύουμε ότι η $(KB \land \neg\alpha)$ είναι μη ικανοποιήσιμη:
  - Εισάγουμε στην KB την $\neg\alpha$.
  - Μετατρέπουμε την $(KB \land \neg\alpha)$ σε μορφή CNF.
  - Εφαρμόζουμε τον κανόνα της ανάλυσης σε οποιοδήποτε ζεύγος προτάσεων όπου μπορεί να εφαρμοστεί.
  - Εάν καταλήξουμε σε άτοπο, η πρόταση α καλύπτεται από την KB.
    - άτοπο σημαίνει κενή πρόταση (false)
  - Ειδάλλως δεν καλύπτεται…

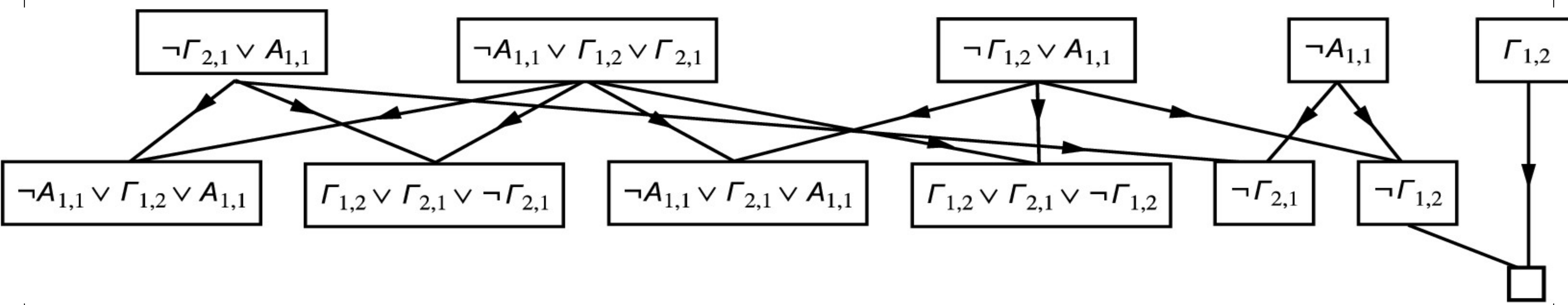
Αλγόριθμος ανάλυσης

```
function PL-RESOLUTION(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
       for each Ci, Cj in clauses do
           resolvents ← PL-RESOLVE(Ci, Cj)
           if resolvents contains the empty clause then return true
           new ← new ∪ resolvents
       if new ⊆ clauses then return false
       clauses ← clauses ∪ new
```

Παράδειγμα

- Έστω οι δύο προτάσεις R_2 και R_4:
  - $KB = R_2 \wedge R_4 = (A_{1,1} \Leftrightarrow (\Gamma_{1,2} \vee \Gamma_{2,1})) \wedge \neg A_{1,1}$
- Θέλουμε να αποδείξουμε την $\neg \Gamma_{1,2}$.
- Μετατρέπουμε την ($KB \wedge \Gamma_{1,2}$) σε CNF και εφαρμόζουμε την ανάλυση:



Αποδοτικός Προτασιακός Συμπερασμός

SAT

Το πρόβλημα εύρεσης ενός μοντέλου για ένα σύνολο προτάσεων στην προτασιακή λογική είναι γνωστό ως ***propositional satisfiability problem*** (PSAT)

Όταν οι προτάσεις είναι σε CNF ονομάζεται **SAT**

Το πρώτο πρόβλημα που αποδείχθηκε ότι είναι NP-complete [Cook 71]

Κάθε διάζευξη από literals ονομάζεται **όρος** (**clause**)

Πολλά πραγματικά προβλήματα μπορούν να αναπαρασταθούν ως SAT προβλήματα

Όταν όλοι οι όροι έχουν 2 literals (2-SAT) το πρόβλημα μπορεί να λυθεί σε πολυωνυμικό χρόνο

Αλγόριθμος DPLL

Davis, Putman, Logemann, Loveland (1962)

Πλήρης αναζήτηση με υπαναχώρηση

Αναδρομική, πρώτα σε βάθος, απαρίθμηση των δυνατών μοντέλων.

Βελτιώσεις

Πρόωρος τερματισμός: Μπορούμε να συμπεράνουμε για την αλήθεια ή το ψεύδος μιας πρότασης, χωρίς να έχουμε τις τιμές όλων των μεταβλητών.

- $(A \lor B) \land (A \lor C)$

Αμιγή σύμβολα: Εμφανίζονται με το ίδιο πρόσημο σε όλες τις προτάσεις.

- $(A \lor \neg B) \land (\neg B \lor \neg C) \land (C \lor A)$

Μοναδιαίες διαζευκτικές προτάσεις: Όλα τα λεκτικά εκτός από ένα είναι *Ψευδή*.

Το πρόβλημα ικανοποιησιμότητας στην προτασιακή λογική είναι γνωστό ως **SAT**

DPLL – Πλήρης Αλγόριθμος για SAT

Davis Putnam Logemann and **Loveland algorithm (DPLL)**

Ανακαλύφθηκε από τους Davis, Logemann and Loveland το 1962

Ο αρχικός SAT αλγόριθμος των Davis & Putnam (αλγόριθμος ανάλυσης) του 1960 χρειάζεται εκθετικό χώρο

Ο DPLL «θυσίασε» χρόνο για χώρο για να δημιουργήσουν έναν αλγόριθμο που απαιτεί μόνο γραμμικό χώρο

DPLL

Αλγόριθμος

Υποθέστε ότι έχουμε ένα σύνολο προτάσεων σε CNF

$$(X \vee Y) \wedge (\neg X \vee Z) \wedge (\neg Y \vee Z) \wedge \ldots$$

Βασική Ιδέα

- Δοκίμασε X=true
- «Διέγραψε» όρους που είναι true

DPLL

Αλγόριθμος

Υποθέστε ότι έχουμε ένα σύνολο προτάσεων σε CNF

$$(\neg X \vee Z) \wedge (\neg Y \vee Z) \wedge \ldots$$

Βασική Ιδέα

- Δοκίμασε X=true
- «Διέγραψε» όρους που είναι true
- Απλοποίησε όρους που περιέχουν το $\neg X$

DPLL

Αλγόριθμος

Υποθέστε ότι έχουμε ένα σύνολο προτάσεων σε CNF

$(\quad Z) \wedge (\neg Y \vee Z) \wedge \ldots$

Βασική Ιδέα

- Δοκίμασε X=true
- «Διέγραψε» όρους που είναι true
- Απλοποίησε όρους που περιέχουν το $\neg X$
- Τώρα μπορούμε να αποφανθούμε ότι το Z πρέπει να είναι true
- Κάποια στιγμή μπορεί να χρειαστεί να οπισθοδρομήσουμε και να δοκιμάσουμε το X=false

DPLL – Βασικές Ιδιότητες

Ο αλγόριθμος DPLL μπορεί να συμπεράνει ότι το πρόβλημα έχει λύση χωρίς να έχει κάνει ανάθεση τιμής σε όλες τις μεταβλητές

$(\neg A \vee \mathrm{B} \vee \mathrm{C}) \wedge (A \vee \mathrm{B} \vee \mathrm{D})$ είναι true αν το B είναι true ανεξάρτητα από τις τιμές των άλλων μεταβλητών

Επίσης ο αλγόριθμος DPLL μπορεί να συμπεράνει ότι το πρόβλημα είναι false χωρίς να έχει κάνει ανάθεση τιμής σε όλες τις μεταβλητές

$(\neg A \vee \mathrm{B} \vee \mathrm{C}) \wedge (A \vee \mathrm{B} \vee \mathrm{D})$ είναι false αν το A είναι true και τα B C είναι false ανεξάρτητα από την τιμή του D

DPLL – Βασικές Ιδιότητες

Unit Clauses (μοναδιαίοι όροι)

Ένας όρος είναι unit όταν περιέχει μόνο ένα literal

Επίσης ένας όρος θεωρείται unit όταν όλα τα literals εκτός από ένα είναι false

- Αν στο πρόβλημα $(\neg A \vee \mathrm{B} \vee \mathrm{C}) \wedge (A \vee \mathrm{B} \vee \mathrm{D})$ το A είναι true και το B είναι false τότε ο όρος $(\neg A \vee \mathrm{B} \vee \mathrm{C})$ είναι unit και προφανώς το C πρέπει να τεθεί σε true

Το unit clause heuristic εντοπίζει όλα τα unit clauses και θέτει τις ελεύθερες μεταβλητές που περιέχουν στην κατάλληλη τιμή ώστε να γίνουν οι όροι true

αυτό μπορεί να δημιουργήσει καινούργια unit clauses

DPLL Αλγόριθμος (απλοποιημένη έκδοση)

```
current_literal = 0;
while (not solution found) and (not insolubility proved)
        if both values of have current_literal been tried
           if current_literal=0
         print "insolubility has been proved";
         break;
     else current_literal--;
        else set current_literal to its next available value;
           if all clauses are true then
        print solution;
        break;
     else if there is no false clause
                 current_literal++;
endwhile
```

DPLL

Βελτιώσεις βασικού αλγόριθμου

heuristics διάταξης μεταβλητών και τιμών

- Ποια θα είναι η επόμενη μεταβλητή που θα πάρει τιμή?
- Ποια τιμή (true/false) θα δοκιμαστεί πρώτα?

Άλλες βελτιώσεις

- Pure literals
- Clause learning
- Έξυπνη οπισθοδρόμηση
- Έξυπνη υλοποίηση

Αλγόριθμοι τοπικής αναζήτησης

Αναρρίχηση λόφων, Προσομοιωμένη ανόπτηση, …

Ευρετική συνάρτηση: Πλήθος διαζευκτικών προτάσεων που δεν ικανοποιούνται.

Αλγόριθμος WalkSat:

Επιλογή τυχαίας διαζευκτικής πρότασης

Επιλογή συμβόλου για αλλαγή τιμής:

- Επιλογή τυχαίου συμβόλου, με πιθανότητα p
- Επιλογή συμβόλου που βελτιστοποιεί την ευρετική συνάρτηση, με πιθανότητα 1-p.

Greedy Local Search: GSAT

Ο GSAT είναι ένας αλγόριθμος αναρρίχησης λόφων για την επίλυση SAT προβλημάτων

Αρχίζοντας με μια αρχική ανάθεση τιμών στις μεταβλητές προσπαθεί να φτάσει σε λύση (ανάθεση τιμών σε μεταβλητές έτσι ώστε να ικανοποιούνται όλοι οι όροι) αλλάζοντας τιμές σε μεταβλητές με βάση μια συνάρτηση αποτίμησης (κόστος)

το κόστος είναι το πλήθος των όρων που είναι FALSE

Ο GSAT δεν είναι πλήρης αλλά σε πολλές περιπτώσεις μπορεί να βρει λύση πολύ γρήγορα

Greedy Local Search: GSAT

Χώρος αναζήτησης S

το σύνολο όλων των πιθανών αναθέσεων τιμών σε όλες τις προτασιακές μεταβλητές

Σύνολο λύσεων S’ $\subseteq$ S

τα μοντέλα της CNF φόρμουλας, δηλ. το σύνολο αναθέσεων τιμών που κάνουν το πρόβλημα TRUE

Διαθέσιμες ενέργειες

αλλαγή της τιμής μιας μεταβλητής

γειτονικές καταστάσεις διαφέρουν στην τιμή μόνο μιας μεταβλητής

Συνάρτηση αποτίμησης f : S -> N

το πλήθος των όρων που δεν ικανοποιούνται (είναι FALSE) με την τρέχουσα ανάθεση

Greedy Local Search: GSAT

- **GSAT:**
  - Ξεκίνα με μια τυχαία ανάθεση τιμών (0 ή 1) στις μεταβλητές
    - Αν όλοι οι όροι είναι TRUE βρέθηκε λύση
  - Άλλαξε την τιμή της μεταβλητής που επιφέρει τη μεγαλύτερη μείωση στο πλήθος των FALSE όρων
  - Επανέλαβε μέχρι όλοι οι όροι να γίνουν TRUE, ή μέχρι να έχουν πραγματοποιηθεί «αρκετές» αλλαγές τιμών μεταβλητών
  - Αν δεν έχει βρεθεί λύση, επανέλαβε τη διαδικασία, αρχίζοντας από διαφορετική αρχική ανάθεση τιμών

A	B	C	$(A \vee C)$	$\wedge$	$(\neg A \vee C)$	$\wedge$	$(B \vee \neg C)$	Score
F	F	F	$\times$		$\surd$		$\surd$	**1**
F	F	T	$\surd$		$\surd$		$\times$	**1**
F	T	T	$\surd$		$\surd$		$\surd$	**0**

(Selman, Levesque, and Mitchell 1992)

GSAT

Παράμετροι του αλγόριθμου:

το όριο των αλλαγών τιμών σε μεταβλητές (max flips)

- όταν ο αλγόριθμος φτάσει αυτό το όριο χωρίς να έχει βρει λύση επαναλαμβάνει τη διαδικασία με διαφορετική αρχική ανάθεση

το όριο των προσπαθειών (max tries) (δηλ. πόσες φορές θα επαναληφθεί η διαδικασία)

- όταν ο αλγόριθμος φτάσει αυτό το όριο χωρίς να έχει βρει λύση, τερματίζει ανεπιτυχώς

ένα σύνολο προτάσεων προτασιακής λογικής σε CNF (φόρμουλα *α*)

GSAT

```
procedure GSAT(a, maxTries, maxFlips)
  for i :=1 to maxTries do
   A := randomly chosen assignment of the variables in a
   for j:=1 to maxFlips do
        if A satisfies a then return (A)
        else
            x := randomly chosen variable of a whose flip satisfies the
  maximum number of clauses under the current assignment A
            if by flipping x you get a cost ≤ current cost then
              flip value of x in A
        endif
   endfor
  endfor
  return ("No model found")
```