Τεχνητή Νοημοσύνη

Επανάληψη

Δρ. Δημήτριος Κουτσομητρόπουλος

ΟΡΙΣΜΟΣ ΤΗΣ ΤΝ;;;

Οι Russell και Norvig διακρίνουν τους ορισμούς της ΤΝ σε τέσσερις κατηγορίες, με βάση το αν χαρακτηρίζουν ένα σύστημα ως ευφυές με κριτήριο το

- **αν σκέφτεται σαν άνθρωπος (Μηχανισμός, Γνωστική Επιστήμη)**
- **αν ενεργεί σαν άνθρωπος (Συμπεριφορά, Turing test)**
- **αν σκέφτεται ορθολογικά (Μηχανισμός, Νόμοι Ορθής Σκέψης)**
- **αν ενεργεί ορθολογικά (Συμπεριφορά, Ορθολογικοί Πράκτορες)**

Επίλυση προβλημάτων με αναζήτηση

Αλγόριθμοι αναζήτησης λύσης

Απληροφόρητοι Αλγόριθμοι Αναζήτησης

- **Breadth-first Search (BFS)**
  Αναζήτηση Πρώτα σε πλάτος
- **Uniform-cost Search (UCS)**
  Αναζήτηση Ομοιόμορφου κόστους
- **Depth-first Search (DFS)**
  Αναζήτηση Πρώτα σε βάθος
- **Iterative deepening Search (IDS)**
  Αναζήτηση επαναληπτικής εκβάθυνσης
- **Bidirectional Search**
  Αμφίδρομη αναζήτηση

Αλγόριθμος Graph-Search

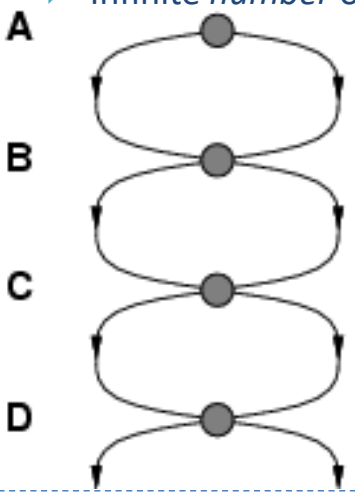
```
function TREE-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        expand the chosen node, adding the resulting nodes to the frontier
```

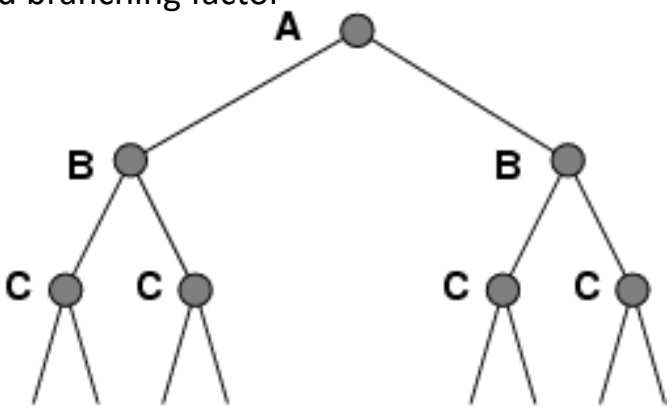
```
function GRAPH-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    initialize the explored set to be empty
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        add the node to the explored set
        expand the chosen node, adding the resulting nodes to the frontier
            only if not in the frontier or explored set
```

- **Μνήμη:** Ο αλγόριθμος θυμάται αυτά που έχει επισκεφθεί
- Explored Set = ***κλειστό σύνολο***

Επαναλαμβανόμενες καταστάσεις

- **Repeated states** (επαναλαμβανόμενες καταστάσεις), προκαλούνται:
  - Redundant paths (πλεονάζοντα μονοπάτια)
  - Loopy paths (κυκλικά μονοπάτια): **Είναι** Redundant, **Είναι** Infinite
- **Infinite Search Tree** (άπειρο δέντρο αναζήτησης), προκαλείται:
  - Infinite paths, go arbitrarily deep
  - Infinite *number* of paths, unbounded branching factor





Ιδιότητες BFS

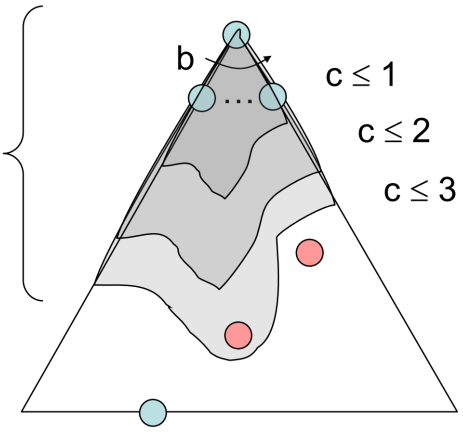
- Πλήρης; Ναι (αν το *b* πεπερασμένο)
  - Κάποια στιγμή θα βρει τον *ρηχότερο* κόμβο-στόχο
- Βέλτιστος; Ναι (*αν το κόστος είναι ίδιο σε κάθε βήμα*)
  - Είναι ο ρηχότερος κόμβος ο βέλτιστος;
  - Κόστος αύξουσα συνάρτηση του βάθους *d*
- Χρόνος; $O(b^d)$
  - Αν η λύση είναι σε βάθος *d* θα δημιουργηθούν και θα ελεγχθούν $1+b+b^2+b^3+\ldots+b^d$ κόμβοι
  - Έλεγχος πριν την επέκταση (αλλιώς: $+b^{d+1}$)
- Χώρος; $O(b^d)$ (*κρατά όλους τους κόμβους στη μνήμη*) *μειονέκτημα*
  - $O(b^d)$ *στο μέτωπο και* $O(b^{d-1})$ στο κλειστό σύνολο

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

Ιδιότητες UCS

- Βέλτιστος; Ναι (*αν το κόστος θετικό σε κάθε βήμα*)
  - Ένας κόμβος προς επέκταση στο μέτωπο βρίσκεται ήδη στο φθηνότερο μονοπάτι του
- Πλήρης; Ναι
  - Εκτός αν υπάρχουν ατέρμονα μονοπάτια μηδενικών βημάτων (*NoOp*)
- Χρόνος; $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
  - C^* το κόστος της βέλτιστης λύσης
  - ε το κόστος του φθηνότερου βήματος
  - C^*/ε το «βάθος» του μονοπατιού στη χειρότερη περίπτωση (Μπορεί >> *d*)
  - Σε κάθε βήμα, έχει επεκτείνει τον γονέα του και έχουν δημιουργηθεί όλα τα αδέρφια του (*b*)
  - Μπορεί να ξαναγυρνά πίσω σε μονοπάτια με πολλά μικρά βήματα
  - Γιατί *+1;* Ποια η σχέση με BFS;
- Χώρος; $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$



Πηγή: ai.berkeley.edu

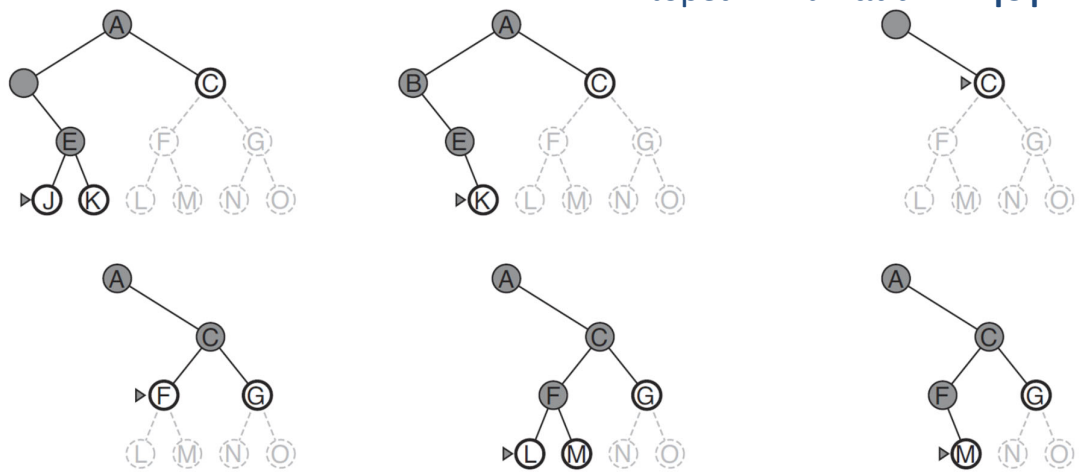
Αναζήτηση πρώτα κατά βάθος (DFS)

Graph-Search εκδοχή

- Πλήρης; **Ναι**
  - Στο τέλος θα επεκτείνει όλους τους κόμβους
- Βέλτιστος; **Όχι**
  - Θα προτιμήσει τη βαθύτερη λύση πρώτα (π.χ. *J* vs. *C*)
- Χρόνος; Το μέγεθος του χώρου καταστάσεων, $|S|$

Tree-Search εκδοχή

- Πλήρης; **Όχι**
  - Μπορεί να εγκλωβιστεί σε κυκλικά μονοπάτια
- Βέλτιστος; **Όχι**
  - Θα προτιμήσει τη βαθύτερη λύση πρώτα (π.χ. *J* vs. *C*)
- Χρόνος; $O(b^m)$
  - m μέγιστο βάθος στο δέντρο. Μπορεί $\boldsymbol{m>>d}$ και $\boldsymbol{b^m >> |S|}$



Αναζήτηση πρώτα κατά βάθος (DFS) – Χώρος

- Graph-Search εκδοχή: Όλοι οι κόμβοι στη μνήμη
  - Όπως και BFS (μέτωπο + κλειστό σύνολο)
- Tree-Search εκδοχή
  - Δεν υπάρχει κλειστό σύνολο!
  - Όταν ένας κόμβος επεκταθεί, αφαιρείται από το μέτωπο
  - Κάθε κόμβος κρατά δείκτη στον γονέα του κοκ μέχρι τη ρίζα
  - Αν όλοι οι απόγονοι αφαιρεθούν, αφαιρείται από τη μνήμη
    - Κανένας δε δείχνει σε αυτόν
  - Χώρος: *O(bm)*
    - *b κόμβοι σε κάθε βήμα του μονοπατιού προς τη ρίζα*
  - Το *m* μπορεί να είναι ***άπειρο***
    - *Κυκλικά μονοπάτια*
    - *Άπειρος χώρος καταστάσεων*

Αναζήτηση περιορισμένου βάθους
(Depth-limited search, DLS)

- Τίθεται όριο βάθους L<m
  - Λύνεται το πρόβλημα των άπειρων διαδρομών
- Αν L<d:
  - Δεν είναι **πλήρης**: Δεν θα φτάσει ποτέ στη λύση
- Αν L>d:
  - Μπορεί να μην είναι **βέλτιστος**: Θα προτιμήσει βαθύτερη λύση
- Χρονός: $O(b^L)$, Χώρος: $O(bL)$

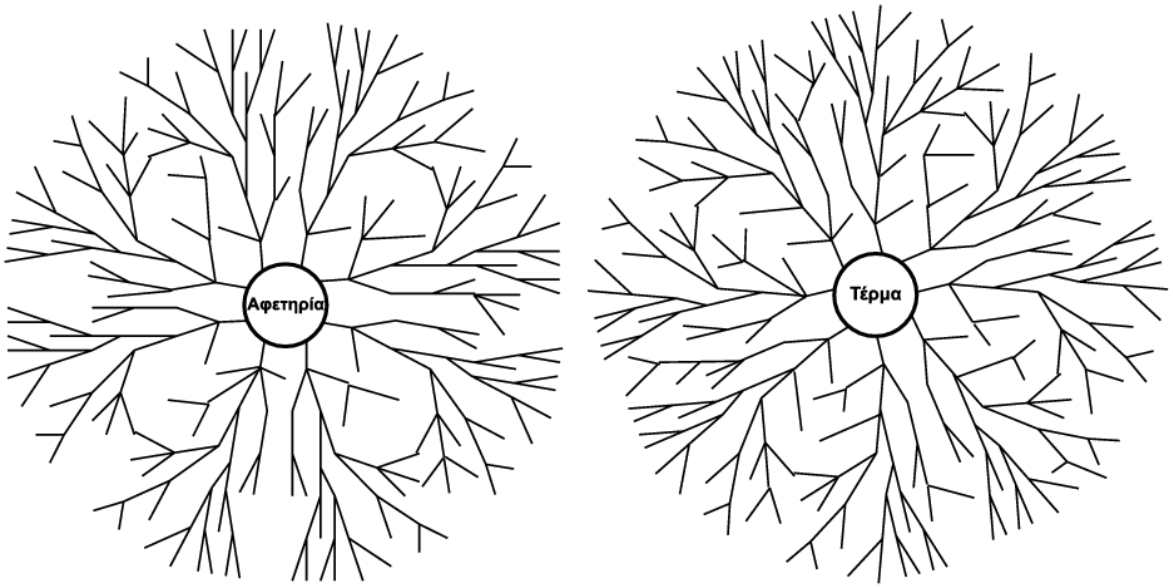
- Επιλογή L με γνώση του προβλήματος (π.χ. διάμετρος του χώρου καταστάσεων)
  - 9 βήματα μέγιστο προς οποιαδήποτε άλλη πόλη

Επαναληπτική εκβάθυνση
(Iterative-deepening search)

- Ιδέα: Σταδιακή αύξηση του ορίου βάθους
  - Πρώτα 0, μετά 1, …., μέχρι ∞
  - Μέχρι να βρει λύση
- Πλεονεκτήματα BFS+DFS μαζί
- Πλήρης: Ναι, εφόσον *b* πεπερασμένο (όπως BFS)
  - Κυκλικά μονοπάτια OK!
- Βέλτιστος: Ναι, εφόσον κόστος ανάλογο του βάθους
  - Θα βρει την *ρηχότερη* λύση (όπως BFS), εξ ορισμού
- Χώρος: $O(bd)$, *d* το βάθος της λύσης (όπως DFS)
- Χρόνος: $O(b^d)$
  Αν η λύση είναι σε βάθος *d* θα τρέξει *d* φορές
  - Πόση σπατάλη γίνεται; Τα μικρότερα βάθη θα δημιουργηθούν πολλές φορές, ενώ το επίπεδο *d* μόνο μία.
  - Οι «πολλοί» κόμβοι είναι στα βαθιά επίπεδα:
  - $N(\text{IDS}) = (d)b + (d-1)b^2 + \ldots + (1)b^d = O(b^d)$

Αμφίδρομη αναζήτηση (1/2)

- Ιδέα: BFS αναζήτηση από την αρχή και το τέλος **ταυτόχρονα**
- Αν τα δύο μέτωπα *τέμνονται* τότε βρέθηκε λύση
- Στη χειρότερη περίπτωση, θα συναντηθούν *στη μέση* (γιατί;)
- $b^{d/2}+b^{d/2} << b^d$
- Επιπλέον (σταθερός) χρόνος αν αυτό είναι το βέλτιστο σημείο τομής



Αμφίδρομη αναζήτηση (2/2)

- Χρονική πολυπλοκότητα: $O(b^{d/2})$
- Χωρική πολυπλοκότητα: $O(b^{d/2})$ (μειονέκτημα)
  - Ακόμα και με IDS, ένα από τα δύο μέτωπα θα πρέπει να παραμένει στη μνήμη για να βρεθεί η κοινή κατάσταση
- Προβλήματα:
  - Εύρεση προκατόχων καταστάσεων (predecessors)
    - Οι τελεστές μετάβασης είναι *αντιστρέψιμοι*;
  - Πώς υλοποιείται αναζήτηση προς τα πίσω από το στόχο;
    - Έλλειψη καλά καθορισμένων στόχων (π.χ. n-queens)

Αλγόριθμοι απληροφόρητης αναζήτησης

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

a) Πλήρης αν b πεπερασμένο
b) Πλήρης αν κόστη βημάτων θετικά
c) Βέλτιστος αν κόστος διαδρομής είναι αύξουσα συνάρτηση του βάθους
d) όταν και οι δύο κατευθύνσεις είναι BFS

Αλγόριθμοι αναζήτησης λύσης

Αλγόριθμοι Πληροφορημένης Αναζήτησης

- **Best-First Search(BestFS)**
  Αναζήτηση Πρώτα στο Καλύτερο
- **Αλγόριθμος A***
  IDA*, MA*/SMA*, **WA***

(Μεθ-) Ευρετικές Μέθοδοι

- **Hill Climbing, Local Search** Αναρρίχηση Λόφων ή Τοπική αναζήτηση
- **Simulated Annealing** Προσομοιωμένη Ανόπτηση
- **Local Beam Search** Τοπική Ακτινική Αναζήτηση
- **Γενετικοί Αλγόριθμοι**
- **Αναζήτηση Tabu**
- ***PSO, ACO, BCO, …***

Πληροφορημένη (ευριστική) αναζήτηση

- **Πληροφορία:** Μια **εκτίμηση** ή **μαντεψιά** του πόσο κοντά στο στόχο βρίσκεται μια κατάσταση
  - Συνάρτηση **αξιολόγησης *f(n)***, *n* μια κατάσταση
    - Σε BFS, DFS *f(n)* το βάθος του κόμβου
  - Μπορεί να εκληφθεί ως *κόστος*
    - *Σε UCS f(n) = g(n),* το κόστος του μονοπατιού μέχρι τον κόμβο.
  - Η *f(n)* περιλαμβάνει μια ***ευρετική συνάρτηση h(n)*** που αναπαριστά αυτήν την πληροφορία
- (greedy) **Best-First Search** (BestFS)
  - Αναζήτηση Πρώτα στο Καλύτερο
  - *h(n)* = εκτίμηση του κόστους του φθηνότερου μονοπατιού από τον κόμβο *n* στο στόχο
  - Στον BestFS απλώς ***f(n) = h(n)***
  - *(Στον A*: f(n) = g(n) + h(n))*

Ιδιότητες BestFS

- Βέλτιστος; Όχι
  - Π.χ. μπορεί να υπάρχει συντομότερο μονοπάτι
  - ***Άπληστος*** αλγόριθμος: τα *άμεσα* φθηνότερα βήματα δεν είναι πάντα τα καλύτερα
- Πλήρης; Ναι – αν Graph-Search
  Όχι – αν Tree-Search, μπορεί να εγκλωβιστεί σε κυκλικά μονοπάτια,
  - Πχ διαδρομή Iasi-Fagaras
  - *Iasi → Neamt → Iasi → Neamt → ….*
- Χρόνος; $O(b^m)$ – χειρότερη περίπτωση
  - *m* το μέγιστο βάθος του χώρου αναζήτησης
  - Μια καλή ευρετική μπορεί να επιφέρει μεγάλη βελτίωση
- Χώρος; $O(b^m)$ – όλοι οι κόμβοι στη μνήμη

Αλγόριθμος αναζήτησης A*

- Ιδέα: Ελαχιστοποίηση του συνολικού *εκτιμώμενου* κόστους
- Παραλλαγή BestFS, UCS
- Συνάρτηση αξιολόγησης *f(n)*:

$$f(n) = g(n) + h(n)$$

- *g(n)*: Το πραγματικό κόστος μέχρι τον κόμβο *n* στη διαδρομή
- *h(n)*: Εκτίμηση του κόστους του φθηνότερου μονοπατιού από τον *n* μέχρι τον στόχο (**π.χ.** h_{SLD})
- Κριτήρια για την ευριστική συνάρτηση h(n):
- **Παραδεκτή** (admissible) (← για Tree-Search)
  - Δεν υπερεκτιμά το κόστος: ∀n, $h(n) \leq h^*(n)$, όπου $h^*(n)$ το πραγματικό κόστος προς τον στόχο από την n.
  - *h(n) κάτω όριο* του πραγματικού κόστους
- **Συνεπής ή μονότονη** (consistent or monotonic) (← για Graph-Search)
  - $h(n) \leq c(n, a, n') + h(n')$, *n΄ διάδοχος του n* **(τριγωνική ανισότητα)**
  - *Αυστηρότερο* κριτήριο που συνεπάγεται το 1ο
- Τότε ο A* είναι και **βέλτιστος και πλήρης**

Ιδιότητες A*

- Βέλτιστος; Ναι
  - Η ευρετική *h(n)* να είναι *παραδεκτή*
  - Η ευρετική *h(n)* να είναι *συνεπής* (χωρίς επανεξέταση κόμβων)
- Πλήρης; Ναι
- Χρόνος; $O(b^d)$ – χειρότερη περίπτωση
  - *d* το βάθος της λύσης (όχι αναγκαστικά της *ρηχότερης*)
  - Π.χ. η ευρετική δεν είναι καλή ή ίδια για όλους ή h(n) = 0 **(UCS)**
  - Μπορεί όμως και *πολυωνυμική*, λόγω κλαδέματος και μη ύπαρξης κύκλων. Εξαρτάται πολύ από την *h(n)*
- Χώρος; $O(b^d)$ – όλοι οι κόμβοι στη μνήμη *μειονέκτημα*
  - Αν Tree-Search; Λιγότερη μνήμη αλλά αύξηση χρόνου
  - Συχνή επέκταση των ίδιων (επαναλαμβανόμενων) καταστάσεων
  - Το πρόβλημα μπορεί να γίνει *δυσεπίλυτο* (intractable)
  - Υπάρχει έντονο *tradeoff* ανάμεσα σε χρόνο/χώρο

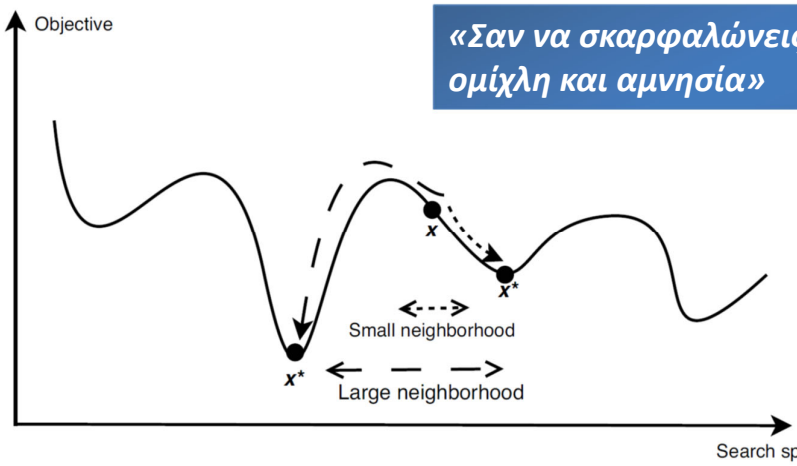
Μεθευρετικές μέθοδοι

- Πρότυπες μεθοδολογίες (templates) για επίλυση γενικευμένων κατηγοριών προβλημάτων, ιδίως βελτιστοποίησης
  - δεν εξαρτώνται από το εκάστοτε πρόβλημα
- Χρησιμοποιούν μια τοπική ευρετική για την δημιουργία νέων λύσεων
  - εξαρτάται από το πρόβλημα
- Δεν είναι ούτε συνεπείς ούτε πλήρεις
  - μπορούν να εντοπίσουν μια ικανοποιητική λύση σε σύντομο χρόνο
- Μπορεί να είναι:
  - *κατασκευαστικοί ή άπληστοι*, χτίζοντας σταδιακά τη λύση σε κάθε βήμα (βλ. BestFS)
  - *επαναληπτικοί*, που ξεκινούν από μια πλήρη λύση (ή πληθυσμό) και τη βελτιώνουν σταδιακά σε κάθε βήμα
    - έχουν ως βάση την **τοπική αναζήτηση (local search)**

Τοπική Αναζήτηση: Η έννοια της γειτονιάς

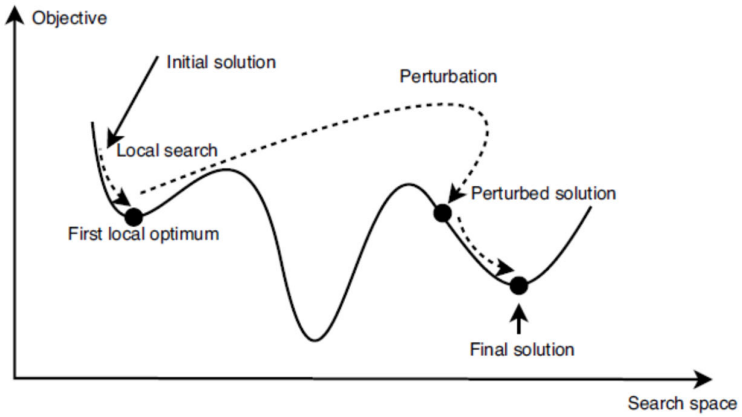
- Γειτονιά της *s*
  - Εξαρτάται από την αναπαράσταση
  - Τοπικότητα (locality): μικρές αλλαγές στην αναπαράσταση επιφέρουν μικρές αλλαγές στη λύση (αλλιώς *τυχαία αναζήτηση*)
  - Συνδετικότητα (connexity): Μπορούμε να φτάσουμε από μία κατάσταση σε οποιαδήποτε άλλη στο χώρο καταστάσεων
  - Τι μέγεθος/ακτίνα έχει; Μικρό μέγεθος για γρήγορη εξερεύνηση, μεγάλο για αποφυγή ***τοπικών βέλτιστων***

«Σαν να σκαρφαλώνεις στο Έβερεστ με πυκνή ομίχλη και αμνησία»



Τυχαία επανεκκίνηση (Random-restart)

- Random restart ή *multistart* local search
  - Σε αποτυχία, η αναζήτηση δε σταματά, αλλά επανεκκινείται από τυχαίο σημείο
  - Σε ορισμένα προβλήματα απαιτούνται πάρα πολλές τέτοιες επανεκκινήσεις μέχρι να βρεθεί (ικανοποιητική) λύση (π.χ. διχοτόμηση γράφου)
  - Ο αριθμός τους αυξάνει εκθετικά με το μέγεθος του προβλήματος
  - Κεντρικό οριακό φαινόμενο:
    Όταν το μέγεθος του προβλήματος γίνει πολύ μεγάλο, τα τοπικά ελάχιστα που προκύπτουν από τυχαίες αρχικές λύσεις είναι παρεμφερή ως προς την ποιότητά τους
- Iterative local search (ILS)
  - Εφαρμόζεται μια ***διατάραξη*** (perturbation) στο τοπικό ελάχιστο και η αναζήτηση επανεκκινείται από εκεί (όχι από τυχαίο σημείο)



Τοπική ακτινική αναζήτηση (Local Beam Search)

- Αποθηκεύονται *k* διαφορετικές καταστάσεις, αντί για μόνο μία
  - Αρχικά *k* καταστάσεις επιλέγονται τυχαία
  - Εκτελείται 1 βήμα local search για κάθε μία
  - Δημιουργείται η γειτονιά τους
  - Επιλέγονται οι *k* καλύτεροι γείτονες *από όλες τις γειτονιές*
  - Συνεχίζεται local search για καθέναν από αυτούς
  - Σύγκριση με random-restart?
- Συχνά οι *k* καλύτεροι τείνουν να συγκεντρώνονται στην ίδια περιοχή (τοπικό ελάχιστο)
  - Απορρίπτονται νωρίς χειρότερες λύσεις που θα μπορούσαν να οδηγήσουν σε βελτίωση αργότερα
  - Εκμετάλλευση vs. εξερεύνηση
- Stochastic local beam search: *k* γείτονες επιλέγονται με πιθανότητα ανάλογη της καταλληλότητας

Αναζήτηση ταμπού (Tabu Search)

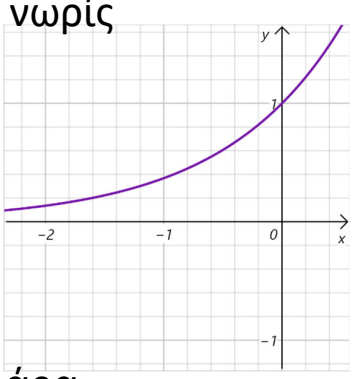
- Ύπαρξη μνήμης για αποθήκευση πληροφοριών σχετικών με τη διαδικασία αναζήτησης
- Όπως στην τοπική αναζήτηση, επιλέγεται ο καλύτερος γείτονας
- Τι γίνεται σε τοπικό βέλτιστο;
  - Η αναζήτηση συνεχίζει επιλέγοντας **χειρότερο** σημείο, δημιουργείται νέα γειτονιά, και επιλέγεται η καλύτερη λύση, ακόμα κι αν είναι χειρότερη από το τοπικό βέλτιστο
  - Μπορεί να δημιουργηθούν κύκλοι (επίσκεψη καταστάσεων που έχουν ήδη δημιουργηθεί)
- Λίστα *Ταμπού* (Tabu list): Αποθηκεύονται οι λύσεις (*ή οι τελεστές*) που έχουν ήδη ελεγχθεί
  - Οι *k* πιο πρόσφατες
  - Όχι ολόκληρες οι λύσεις, αλλά χαρακτηριστικά τους (attributes)
  - Π.χ. μια λίστα ακμών για το TSP

Προσομοιωμένη Ανόπτηση (Simulated Annealing)

- Εμπνευσμένη από τη μεταλλουργία
  - **Ανόπτηση:** *Θέρμανση σε υψηλή θερμοκρασία και μετά σταδιακή ψύξη με συγκεκριμένο ρυθμό, ώστε το υλικό να αποκτήσει βέλτιστες ιδιότητες (σκλήρυνση)*
- Χειρότερες λύσεις μπορεί να γίνουν αποδεκτές από νωρίς
  - Επιλέγεται τυχαία μια γειτονική λύση
  - Αν είναι καλύτερη θα γίνει αποδεκτή
  - Αν είναι χειρότερη ($f(x')>f(x)$), θα γίνει αποδεκτή με πιθανότητα p:

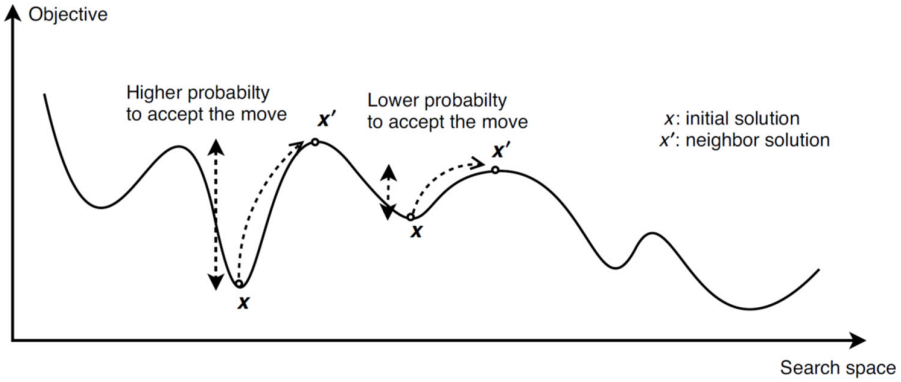
$$p = e^{-\frac{f(x')-f(x)}{T}}$$

  - x': γείτονας, T: θερμοκρασία



- Όσο πιο μεγάλο το Τ, τόσο πιο μικρός ο εκθέτης και άρα πιο κοντά στη μονάδα (αφού είναι πάντα αρνητικός).
  - Χειρότερες λύσεις θα επιλέγονται αρκετά συχνά
- Όσο μικραίνει το Τ, μεγαλώνει ο εκθέτης και απομακρυνόμαστε από την μονάδα προς το 0.
  - Χειρότερες λύσεις θα γίνουν πιο σπάνια αποδεκτές

Προσομοιωμένη Ανόπτηση (Simulated Annealing)



- Σύγκλιση στη βέλτιστη λύση!
  - Αν η θερμοκρασία μειώνεται αρκετά αργά, ο αλγόριθμος θα βρει το **ολικό βέλτιστο** με πιθανότητα → 1
  - Όμως, *ασυμπτωτικά* (σύγκλιση μετά από άπειρα βήματα)
- Επιλογές παραμέτρων
  - Τι αρχική θερμοκρασία;
  - Τι πρόγραμμα ανόπτησης (ψύξης);
- Πολύ καλά (near optimal) αποτελέσματα για TSP, VLSI

Γενετικοί Αλγόριθμοι

- Εμπνευσμένοι από τη θεωρία της εξέλιξης και της φυσικής επιλογής
  - *οι οργανισμοί που είναι περισσότερο προσαρμοσμένοι στο περιβάλλον τους επιβιώνουν και αναπαράγονται περισσότερο από τους λιγότερο προσαρμοσμένους*
- Μια διάδοχη κατάσταση προκύπτει συνδυάζοντας δύο καταστάσεις-γονείς
- Ξεκινά με *k* τυχαία επιλεγμένες καταστάσεις **(πληθυσμός)**
- Μια κατάσταση αναπαρίσταται ως *συμβολοσειρά* (0 και 1, ακέραιοι αριθμοί, κτλ)
- **Συνάρτηση καταλληλότητας:** Καλύτερες τιμές για καλύτερες λύσεις.
- Παραγωγή της επόμενης γενιάς χρησιμοποιώντας γενετικούς τελεστές
  - **Επιλογή** (selection)
  - **Διασταύρωση** (crossover)
  - **Μετάλλαξη** (mutation)
- Οι τελεστές εφαρμόζονται *στοχαστικά*

Ικανοποίηση Περιορισμών

Αναπαράσταση Προβλήματος

- Ένα πρόβλημα ικανοποίησης περιορισμών (constraint satisfaction problem) ορίζεται από:
  - Ένα σύνολο *n* μεταβλητών $V_1, V_2,...,V_n$,
  - Ένα σύνολο *n* πεδίων τιμών $D_1, ...D_n$, που αντιστοιχούν σε κάθε μεταβλητή έτσι ώστε $V_i \in D_i$, και
  - Ένα σύνολο σχέσεων (περιορισμών) $C_1, C_2,...C_m$ όπου $C_i(V_k,...,V_r)$ μια σχέση μεταξύ μεταβλητών του προβλήματος.
- Ανάλογα με το πόσες μεταβλητές περιλαμβάνει ένας περιορισμός χαρακτηρίζεται ως *μοναδιαίος* (*unary*), *δυαδικός* (*binary*) ή *ανώτερης* τάξης (*higher order*)

Λύση Προβλήματος Περιορισμών

- Λύση αποτελεί μια ανάθεση τιμών στις μεταβλητές του προβλήματος:

  $V_1 = d_1, V_2 = d_2, \ldots V_n = d_n$ (όπου $d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n$)

  τέτοια ώστε να ικανοποιούνται οι περιορισμοί $C_1, C_2, \ldots, C_m$
- Τα προβλήματα με D_i διακριτών τιμών αναφέρονται ως *προβλήματα ικανοποίησης περιορισμών*, ενώ αυτά με D_i συνεχών τιμών ως *προβλήματα επίλυσης περιορισμών*.
- Η αναζήτηση λύσης στα προβλήματα ικανοποίησης περιορισμών οδηγεί στο φαινόμενο της συνδυαστικής έκρηξης (combinatorial explosion).
- Γι' αυτό απαιτούνται ειδικοί αλγόριθμοι **για τη μείωση του χώρου αναζήτησης.**

Κατηγορίες Αλγορίθμων Συνέπειας

- **Αλγόριθμος συνέπειας κόμβου (Node Consistency)**
  - Μοναδιαίοι περιορισμοί (αφαιρεί τιμές πεδίων βασισμένος σε αυτούς).
- **Αλγόριθμοι συνέπειας τόξου (Arc Consistency-AC)**
  - Δυαδικοί περιορισμοί
  - Διάφοροι αλγόριθμοι συνέπειας τόξου, όπως οι AC-3, AC-4, AC-5, AC-6, κλπ.
  - Η δυσκολία που παρουσιάζουν οι αλγόριθμοι της κατηγορίας:
    - Διαγραφή μιας τιμής οδηγεί σε αλλαγές στα πεδία άλλων μεταβλητών.
    - Μετά από κάθε διαγραφή ασυνεπούς τιμής πρέπει να επανεξεταστούν τα πεδία των "άμεσα" συνδεδεμένων μεταβλητών.
- **Αλγόριθμοι συνέπειας μονοπατιού (path consistency algorithms)**
  - Περιορισμοί υψηλότερης τάξης
  - Υψηλό υπολογιστικό κόστος.

K-ΣΥΝΕΠΕΙΑ

Ένα πρόβλημα ικανοποίησης περιορισμών έχει **k-συνέπεια** αν, για οποιοδήποτε σύνολο $k-1$ μεταβλητών και για οποιαδήποτε συνεπή ανάθεση τιμών σε αυτές τις μεταβλητές, μια συνεπής τιμή μπορεί πάντα να ανατεθεί σε οποιαδήποτε k-οστή μεταβλητή.

Ένα πρόβλημα ικανοποίησης περιορισμών έχει **ισχυρή k-συνέπεια** (strong k-consistency) αν έχει k-συνέπεια και έχει επίσης $(k-1)$-συνέπεια, $(k-2)$-συνέπεια,… μέχρι και 1-συνέπεια.

- Ο αλγόριθμος συνέπειας κόμβου εξασφαλίζει ότι ο γράφος είναι ισχυρά 1-συνεπής.
- Οι αλγόριθμοι συνέπειας τόξου εξασφαλίζουν ισχυρή 2-συνεπεία.
- Προφανώς σε ένα γράφο με n κόμβους, εάν εξασφαλισθεί ότι ο γράφος είναι ισχυρά n-συνεπής, τότε
  - Μπορεί να βρεθεί λύση χωρίς αναζήτηση.
  - Ο συνολικός χρόνος εκτέλεσης είναι μόνο $O(n^2 d)$, όπου d το πλήθος των τιμών στο πεδίο κάθε μεταβλητής.
  - Όμως σε προβλήματα με k>2 το υπολογιστικό κόστος εφαρμογής είναι υψηλό, οπότε προτιμάται ο συνδυασμός το πολύ 2-συνέπειας με κλασσικό αλγόριθμο αναζήτησης.

Ο αλγόριθμος AC-3

- Ο απλούστερος αλγόριθμος συνέπειας τόξου.
- Έστω οι μεταβλητές V_1, V_2,..V_n με τιμές d_1, d_2,…,d_n από τα πεδία τιμών των μεταβλητών D_1, D_2,…,D_n και ένα σύνολο περιορισμών $C(V_i,V_j)$ για τις μεταβλητές αυτές, οι οποίοι αναπαριστώνται ως τόξα (V_i,V_j). Για συντομία, κάθε τόξο (V_i,V_j) αναφέρεται ως (i,j).
- Το Q περιλαμβάνει αρχικά όλα τα τόξα του γράφου περιορισμών.

Επανέλαβε τα ακόλουθα βήματα μέχρι το Q να γίνει κενό:

1. **Επέλεξε ένα τόξο (i,j) και διέγραψε το από το Q**
2. **Για κάθε τιμή d_i του πεδίου της μεταβλητής V_i έλεγξε αν υπάρχει τουλάχιστον μία τιμή d_j του πεδίου της μεταβλητής V_j τέτοια ώστε να ικανοποιεί το περιορισμό $C(V_i,V_j)$ που αντιστοιχεί στο τοξο (i, j).**
3. **Αν δεν υπάρχει τέτοια τιμή d_i τότε αφαίρεσε την τιμή d_i από το πεδίο τιμών της V_i. Αν το πεδίο τιμών της V_i είναι κενό τότε τερμάτισε με αποτυχία.**
4. **Αν έχει μεταβληθεί το πεδίο τιμών της V_i τότε πρόσθεσε στο σύνολο Q όλα τα τόξα (k,i), που αντιστοιχούν στους περιορισμούς $C(V_k,V_i)$, για $k \neq i$.**

Συνδυασμός Αλγορίθμων Συνέπειας και Κλασικής Αναζήτησης

- ***Ο προοπτικός έλεγχος (Forward checking)***
  - Απαλείφει τιμές από τα πεδία των μη-δεσμευμένων μεταβλητών που συνδέονται άμεσα με περιορισμούς με την **(μία)** μεταβλητή στην οποία μόλις ανατέθηκε τιμή.
    - Παραμένει μεγάλος αριθμός ασυνεπών τιμών στα πεδία.
    - Χαμηλό υπολογιστικό κόστος κάθε βήματος.
- ***Ο αλγόριθμος έγκαιρης μερικής εξέτασης (Partial Look Ahead)***
  - Κατευθυντική συνέπεια (directional consistency) σε κάθε βήμα (εξετάζει όλα τα πεδία τιμών των μη-δεσμευμένων μεταβλητών με προκαθορισμένη σειρά, ελέγχοντας τους περιορισμούς μία μόνο φορά).
    - Παραμένουν στα πεδία των μεταβλητών μη συνεπείς τιμές.
- ***Ο αλγόριθμος έγκαιρης πλήρους εξέτασης (Full Look Ahead) ή διατήρησης συνέπειας τόξου (Maintaining Arc Consistency - MAC).***
  - Εφαρμόζει πλήρη αλγόριθμο συνέπειας τόξου σε κάθε βήμα.
    - Αφαιρεί το μεγαλύτερο αριθμό ασυνεπών τιμών από τους τρεις.
    - Υψηλό υπολογιστικό κόστος

Αναπαράσταση Γνώσης και Συλλογισμός

Λογική ως Γλώσσα Αναπαράστασης Γνώσης - Θεωρία Μοντέλων

- **Ερμηνεία** (Interpretation): $\boldsymbol{I} = < \boldsymbol{D}, f_I >$
  - **D**: Σύνολο πρωτογενών οντοτήτων
  - f_I: Ερμηνευτική συνάρτηση
    - συσχετίζει σύμβολα με οντότητες
- **Μοντέλο πρότασης** (model): $\boldsymbol{I} \vDash \varphi$
  - φ αληθής με βάση την **I**
  - Η **I** ικανοποιεί την φ ή η **I** είναι μοντέλο της φ
- **Μοντέλο συνόλου προτάσεων:**
  - **S** σύνολο προτάσεων
  - **I** μοντέλο **S** ανν $\forall \varphi \in \boldsymbol{S}, \boldsymbol{I}$ μοντέλο φ

- Κάθε μοντέλο περιλαμβάνει μια **ερμηνεία** (interpretation) που καθορίζει ακριβώς σε ποια αντικείμενα, σχέσεις και συναρτήσεις αναφέρονται τα σύμβολα των σταθερών, των κατηγορημάτων και των συναρτήσεων.
  - ***Επιδιωκόμενη ερμηνεία*** *(intended interpretation):* Η ερμηνεία που είχαμε στο μυαλό μας όταν επιλέγαμε ονόματα συμβόλων.

Λογική ως ΓΑΓ - Θεωρία Μοντέλων

- **Λογική συνεπαγωγή** (logical implication)
  - Από πρόταση

$$\varphi_1 \vDash \varphi_2 \; \alpha\nu\nu \; \forall \boldsymbol{I} : \; \boldsymbol{I} \vDash \varphi_1 \Rightarrow \boldsymbol{I} \vDash \varphi_2$$

    - Ιδιότητες: ανακλαστική, μεταβατική
  - Από σύνολο προτάσεων

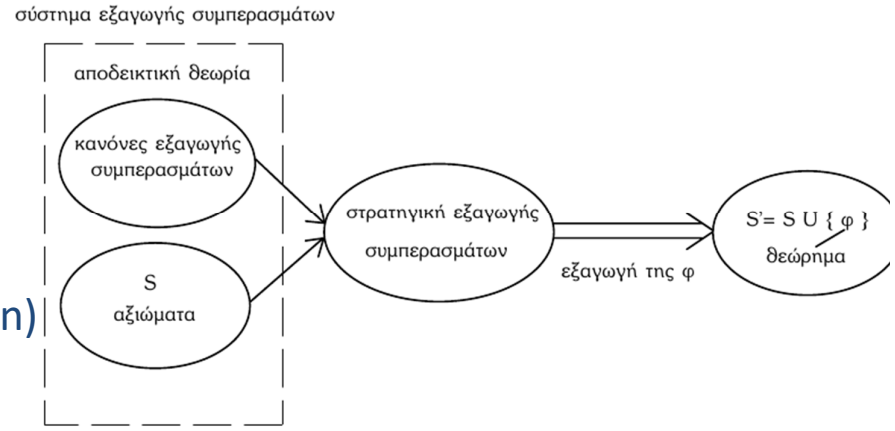
$$\boldsymbol{S} \vDash \varphi' \; \alpha\nu\nu \; \forall \varphi \in \boldsymbol{S}, \forall \boldsymbol{I} : \; \boldsymbol{I} \vDash \varphi \Rightarrow \boldsymbol{I} \vDash \varphi'$$

  - Άλλη ορολογία:
    - Έγκυρο επακόλουθο (valid consequence)
    - Λογική συνέπεια (logical consequence)
    - Σημαντική συνέπεια (semantic consequence)
- **Λογική ισοδυναμία**

$$\varphi_1 \equiv \varphi_2 \; \alpha\nu\nu \; \varphi_1 \vDash \varphi_2 \; \kappa\alpha\iota \; \varphi_2 \vDash \varphi_1$$

Αποδεικτική Θεωρία

- **Απόδειξη** (proof) πρότασης φ από $\boldsymbol{S}$
  - Μια ακολουθία προτάσεων με τελευταία τη φ και κάθε άλλη είτε από το $\boldsymbol{S}$ είτε εξαγχθείσα από το $\boldsymbol{S}$
  - Άλλη ορολογία:
  - **Συνεπαγωγή** (deduction)
  - **Εξαγωγή** (derivation)



ΚΑΤΗΓΟΡΗΜΑΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΠΡΩΤΗΣ ΤΑΞΗΣ - ΚΛΠΤ (FIRST ORDER PREDICATE CALCULUS - FOPC)

- **Πεδίο ορισμού**: $\boldsymbol{D}$ (το σύνολο των αντικειμένων που θέλουμε να αναπαραστήσουμε)
- **Σταθερές** (μία τιμή από το $\boldsymbol{D}$): $\{C_i \in \boldsymbol{D}\}$
- **Λογικές σταθερές**: $\{T, F\}$
- **Μεταβλητές** (υποσύνολο του D): $\{v_i \subseteq \boldsymbol{D}\}$
- **Κατηγορήματα**: $\{P_i^n\}$: $\boldsymbol{D}^n \rightarrow \{T, F\}$, εκφράζουν σχέσεις ή ιδιότητες μεταξύ *n* αντικειμένων
- **Συναρτήσεις**: $\{f_i^n\}$: $\boldsymbol{D}^n \rightarrow \boldsymbol{D}$, αντιστοιχίζουν *n* αντικείμενα σε ένα άλλο

- Δεν χρειάζεται όλα τα αντικείμενα (π.χ., τα αντικείμενα που αποτελούν αποτέλεσμα συνάρτησης) να έχουν όνομα
- Είναι επίσης δυνατό ένα αντικείμενο να έχει πολλά ονόματα.

ΚΛΠΤ - Γραμματική

$$
\begin{aligned}
\textit{Πρόταση} &\rightarrow \textit{ΑτομικήΠρόταση} \mid \textit{ΣύνθετηΠρόταση} \\
\textit{ΑτομικήΠρόταση} &\rightarrow \textit{Κατηγόρημα} \mid \textit{Κατηγόρημα}(\textit{Όρος}, \ldots) \mid \textit{Όρος} = \textit{Όρος} \\
\textit{ΣύνθετηΠρόταση} &\rightarrow (\textit{Πρόταση}) \\
&\mid \neg \textit{Πρόταση} \\
&\mid \textit{Πρόταση} \wedge \textit{Πρόταση} \\
&\mid \textit{Πρόταση} \vee \textit{Πρόταση} \\
&\mid \textit{Πρόταση} \Rightarrow \textit{Πρόταση} \\
&\mid \textit{Πρόταση} \Leftrightarrow \textit{Πρόταση} \\
&\mid \textit{Ποσοδείκτης Μεταβλητή}, \ldots \textit{Πρόταση} \\
\textit{Όρος} &\rightarrow \textit{Συνάρτηση}(\textit{Όρος}, \ldots) \\
&\mid \textit{Σταθερά} \\
&\mid \textit{Μεταβλητή} \\
\textit{Ποσοδείκτης} &\rightarrow \forall \mid \exists \\
\textit{Σταθερά} &\rightarrow A \mid X_1 \mid \textit{Ιωάννης} \mid \cdots \\
\textit{Μεταβλητή} &\rightarrow a \mid x \mid s \mid \cdots \\
\textit{Κατηγόρημα} &\rightarrow \textit{Αληθές} \mid \textit{Ψευδές} \mid \textit{Μετά} \mid \textit{Αγαπά} \mid \textit{Βρέχει} \mid \cdots \\
\textit{Συνάρτηση} &\rightarrow \textit{Μητέρα} \mid \textit{ΑριστερόΠόδι} \mid \cdots \\
\text{Προτεραιοτητα τελεστων} &: \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Έλεγχος μοντέλων στην Προτασιακή Λογική

- Ο στόχος μας είναι να αποφασίσουμε αν $KB \models a$ για κάποια πρόταση a.
- **Έλεγχος μοντέλων** (model checking): απαριθμεί όλα τα δυνατά μοντέλα για να ελέγξει ότι η a είναι αληθής σε όλα τα μοντέλα στα οποία η βάση γνώσης KB είναι αληθής
- Μπορεί να γίνει με έλεγχο μοντέλων και αναδρομική αναζήτηση πρώτα σε βάθος (Backtracking-Search).
  - Ουσιαστικά είναι σαν να επιλύουμε πρόβλημα ικανοποίησης περιορισμών με αναζήτηση πρώτα σε βάθος, χωρίς όμως έλεγχο συνέπειας.

Έλεγχος μοντέλων στην Προτασιακή Λογική (2)

- Αν οι προτάσεις KB και a περιέχουν n σύμβολα συνολικά, τότε υπάρχουν 2^n μοντέλα.
- Επομένως η **χρονική πολυπλοκότητα του αλγορίθμου είναι $O(2^n)$.**
  - Η χωρική πολυπλοκότητα είναι μόνο $O(n)$, επειδή η απαρίθμηση γίνεται πρώτα σε βάθος.
- Η ***προτασιακή*** λογική είναι coNP-complete
  - μάλλον δεν είναι ευκολότερη από ό,τι αν ήταν NP-πλήρης
  - επομένως κάθε γνωστός αλγόριθμος συμπερασμού για την προτασιακή λογική έχει πολυπλοκότητα χειρότερης περίπτωσης που είναι εκθετική ως προς το μέγεθος της εισόδου.

Απόδειξη θεωρημάτων

- Είναι δυνατή η εφαρμογή κανόνων συμπερασμού απευθείας στις προτάσεις της βάσης γνώσης, για την κατασκευή μιας απόδειξης για την επιθυμητή πρόταση, χωρίς να λάβουμε υπόψη μας τα μοντέλα.
  - Αν το πλήθος των μοντέλων είναι μεγάλο αλλά το μήκος της απόδειξης είναι μικρό, τότε η **απόδειξη θεωρημάτων** μπορεί να είναι πιο αποδοτική από τον έλεγχο μοντέλων.
- **Οριστική πρόταση** (definite clause) χαρακτηρίζεται μια διάζευξη λεκτικών από τα οποία ακριβώς ένα είναι θετικό.
  - Για παράδειγμα, η διαζευκτική πρόταση $(\neg L_{1,1} \vee \neg \text{Αύρα} \vee B_{1,1})$ είναι οριστική πρόταση, ενώ η $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$ δεν είναι, επειδή έχει δύο θετικά λεκτικά.
- **Πρόταση Horn** (Horn clause) χαρακτηρίζεται μια διάζευξη λεκτικών από τα οποία το πολύ ένα είναι θετικό.
  - Όλες οι οριστικές προτάσεις είναι προτάσεις Horn.
  - Οι προτάσεις χωρίς θετικά λεκτικά ονομάζονται **προτάσεις στόχου** (goal clauses)
  - Οι προτάσεις Horn δεν έχουν πλήρη εκφραστικότητα στην προτασιακή λογική.
  - Ο προσδιορισμός της λογικής συνεπαγωγής με προτάσεις Horn μπορεί να γίνεται σε **χρόνο γραμμικό ως προς το μέγεθος της βάσης γνώσης** (forward-, backward-chaining)

Αναγωγή στον προτασιακό συμπερασμό

- Γενική τεχνική **μετατροπής** ΚΛΠΤ σε προτασιακή μορφή (propositionalization)
- **Πρόβλημα:** όταν η βάση γνώσης περιλαμβάνει ένα συναρτησιακό σύμβολο, καθώς το σύνολο των δυνατών αντικαταστάσεων βασικών όρων είναι άπειρο!
- **Θεώρημα Herbrand (1930)**: *αν μια πρόταση συνεπάγεται από την αρχική βάση γνώσης ΚΛΠΤ*, τότε υπάρχει απόδειξη που αφορά **μόνο ένα πεπερασμένο υποσύνολο** της προτασιακής βάσης γνώσης.
- Όμως: *Αν η πρόταση δεν συνεπάγεται από τη βάση γνώσης*, αυτό δεν μπορεί να αποδειχθεί **(Church-Turing thesis, 1936)**
- ΚΛΠΤ είναι ***ημιαποφασίσιμη*** *(semidecidable)* – δηλαδή, ενώ υπάρχουν αλγόριθμοι που απαντούν θετικά σε κάθε συνεπαγόμενη πρόταση, δεν υπάρχει κάποιος αλγόριθμος που να απαντά αρνητικά σε κάθε μη συνεπαγόμενη πρόταση.
- Αλγόριθμοι συνεπείς (sound), αλλά όχι πλήρεις (complete)

Αν $KB \not\models Q(x)$ τότε $KB \wedge \neg Q(x)$ ικανοποιήσιμη

  - Πλήρεις ως προς την ***αντίφαση ή διάψευση*** (refutation complete)
  - πλήρεις για KB με ***προτάσεις Horn*** χωρίς συναρτησιακά σύμβολα *(Datalog).*

ΚΛΠΤ - Κανονικές Μορφές

- **Συζευκτική Κανονική Μορφή - ΣΚΜ** (Conjunctive Normal Form - CNF)

$$(H \vee G) \wedge (\neg A \vee G) \wedge \dots$$

  - Σε μορφή CNF, τα λεκτικά μπορούν να περιέχουν μεταβλητές, οι οποίες θεωρούνται ότι είναι καθολικά ποσοτικοποιημένες (∀, ισχύει *για όλα*)

Κάθε πρόταση λογικής πρώτης τάξης μπορεί να μετατραπεί σε μια ***ισοδύναμη όσον αφορά τον συμπερασμό*** πρόταση CNF.

- **Διαζευκτική Κανονική Μορφή - ΔΚΜ** (Disjunctive Normal Form - DNF)

$$(H \wedge G) \vee (\neg A \wedge G) \vee \dots$$

- **Κανονική Μορφή Prenex - ΚΜΡ** (Prenex Normal Form - PNF)

$$(Q_1 x_1)(Q_2 x_2) \dots (Q_n x_n)(H)$$

H, G, A ΚΣΕς
Q_i ποσοδείκτες

Μετατροπή ΚΛΠΤ σε Προτασιακή Μορφή

Διαδικασία:

1. Απαλοιφή συνεπαγωγών
2. Περιορισμός εμβέλειας αρνήσεων
3. Μετονομασία μεταβλητών με το ίδιο όνομα που δεσμεύονται από διαφορετικούς ποσοδείκτες
4. Μετατροπή σε ΚΜΡ (PNF)
5. Απαλοιφή υπαρξιακών ποσοδεικτών (Skolemisation)
   - **Σταθερές Skolem**: Αντικαθιστούμε τη μεταβλητή με ένα ***νέο όνομα***
   - **Συναρτήσεις Skolem**: Αντικαθιστούμε τη μεταβλητή με μια ***νέα συνάρτηση***, ώστε να εξαρτάται από τις άλλες μεταβλητές
     - Όλες τις καθολικά ποσοτικοποιημένες μεταβλητές στην εμβέλεια των οποίων εμφανίζεται ο υπαρξιακός ποσοδείκτης.
     - Η πρόταση Skolem ικανοποιείται ακριβώς όταν ικανοποιείται και η αρχική πρόταση.

Αρχή της Επίλυσης (Resolution Principle)

- Είναι ένας **κανόνας εξαγωγής συμπερασμάτων** (ΚΕΣ) που εφαρμόζεται στην προτασιακή μορφή του ΚΛΠΤ
- Αναφέρεται στην παραγωγή μιας "νέας" πρότασης από δύο υπάρχουσες
- Επειδή από μόνος του ο κανόνας αυτός **δεν εξασφαλίζει πληρότητα**, συνοδεύεται συνήθως από ένα απλούστερο κανόνα (ή μετασχηματισμό), την παραγοντοποίηση (factoring).
- Η παραγοντοποίηση επιδρά σε μια πρόταση και την μετασχηματίζει σε μια άλλη, στηριζόμενη στην ενοποίηση στοιχείων της πρότασης

Αρχή της Επίλυσης (Resolution Principle) (2)

- **Παράγων (factor):** Αν δύο ή περισσότερα στοιχεία μιας πρότασης C (ίδιας πολικότητας) έχουν μια γενικότερη ενοποιήτρια γ τότε η $C\gamma$ καλείται παράγων της C
  - Η ***παραγοντοποίηση*** απλοποιεί την πρόταση, απαλείφοντας στοιχεία που ενοποιούνται

- **Αρχή της Επίλυσης** (ΑΕ): Αν L_1, L_2 είναι στοιχεία των C_1, C_2 αντίστοιχα και τα $L_1, \neg\boldsymbol{L_2}$ έχουν μια γενικότερη ενοποιήτρια σ τότε η $(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$ καλείται <u>δυαδική επιλύουσα</u> (binary resolvent) των C_1, C_2

- **<u>Παράδειγμα:</u>** $C_1 = \{p(x), q(x)\}$ και $C_2 = \{\neg p(a), r(y)\}$
  Αν επιλέξουμε, $L_1 = p(x), L_2 = \neg p(a)$ με γ.ε. $\sigma = \{^a/_x\}$
  τότε, η $R = \{q(a), r(y)\}$ είναι επιλύουσα των C_1, C_2

Απόδειξη Θεωρημάτων (Theorem Proving)

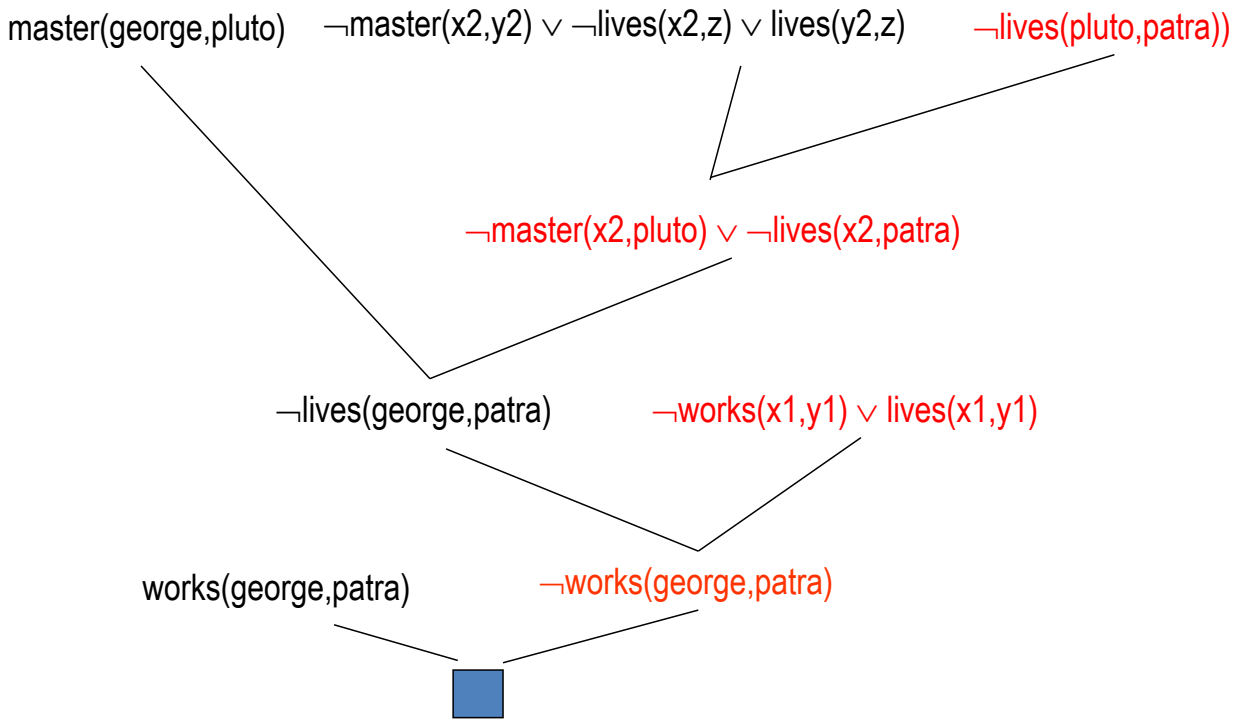
- **Θεώρημα:** Αν $S \cup \{\varphi\}$ είναι ασυνεπές τότε $S \models \neg\varphi$. Άρα αν $S \cup \{\neg\varphi\}$ είναι ασυνεπές τότε $S \models \varphi$, όπου S είναι ένα σύνολο λογικών προτάσεων

<u>ΑΝΤΙΦΑΣΗ ΕΠΙΛΥΣΗΣ (RESOLUTION REFUTATION)</u>

Θέλουμε να αποδείξουμε την φ αν έχουμε το σύνολο προτάσεων S

1. $S' = S \cup \{\neg\varphi\}$
2. Εφαρμογή της Αρχής της Επίλυσης, παραγωγή επιλύουσας C
3. Αν $C =$ κενή πρόταση, σταμάτα (επιτυχία)
4. $S' = S \cup \{C\}$
5. Πήγαινε στο 2

Απόδειξη Θεωρημάτων - Παράδειγμα



Στρατηγικές Ελέγχου της Επίλυσης Επιλογής Γονέα: Γραμμική Επίλυση - Παραλλαγές

- **LI-Επίλυση** (Γραμμική Επίλυση Εισόδου/Linear Input Resolution)
  - Ο ένας γονέας περιορίζεται να είναι μόνο αξίωμα (πρόταση εισόδου), ενώ ο κοντινός γονέας η πιο πρόσφατη επιλύουσα
  - Πλήρης μόνο για προτάσεις τύπου Horn
- **LD-Επίλυση** (Γραμμική Ορισμένη Επίλυση/Linear Definite Resolution)
  - Γραμμική επίλυση εισόδου, όπου οι προτάσεις θεωρούνται διατεταγμένα σύνολα (ακολουθίες) και η παραγωγή της επιλύουσας γίνεται κατά συγκεκριμένο τρόπο.
  - Πλήρης μόνο για προτάσεις τύπου Horn
- **SLD-Επίλυση** (Selection Linear Definite Resolution)
  - Γραμμική ορισμένη επίλυση, με έναν κανόνα επιλογής που καθορίζει ποιο στοιχείο της πρότασης στόχου κάθε φορά εξετάζεται προς επίλυση
  - Πλήρης μόνο για προτάσεις τύπου Horn
  - **Συνήθης κανόνας:** επιλέγεται το πρώτο αριστερά στοιχείο.
  - Η βάση της στρατηγικής του ***λογικού προγραμματισμού***

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Η απόδειξη ενός στόχου γίνεται με τη χρήση του κανόνα **SLD-Επίλυση** (SLD-Resolution principle):

$$\frac{(\leftarrow A_1, \dots, A_{i-1}, A_i, A_{i+1}, \dots, A_n) \quad (B_o \leftarrow B_1, B_2, \dots, B_m)}{(\leftarrow A_1, \dots, A_{i-1}, B_1, B_2, \dots, B_m, A_{i+1}, \dots, A_n)\theta}$$

όπου $\theta = \text{γ.ε.}(A_i, B_o)$

Η SLD-Επίλυση είναι ***refutation complete*** μόνο για Horn clauses!

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

- Τα οριστικά προγράμματα εκφράζουν μόνο θετική γνώση (πώς σχετίζονται μεταξύ τους οντότητες του πεδίου του προβλήματος)
- Δεν εκφράζουν αρνητική γνώση (τι δεν ισχύει)
- Για να υπερβούμε το εμπόδιο αυτό χρησιμοποιείται η λεγόμενη **«υπόθεση κλειστού κόσμου» (closed world assumption-CWA)**
- $KB^{+} \rightarrow \{ KB \cup \neg f \mid KB \nvdash f \}$
- Η cwa είναι μια αρχή που αναφέρει ότι αν ένα θετικό στοιχείο Α χωρίς μεταβλητές δεν μπορεί να αποδειχθεί από ένα οριστικό πρόγραμμα τότε συμπεραίνουμε το ¬Α.
- Η KB μπορεί να γίνει εύκολα ***ασυνεπής*** για μη Horn προτάσεις, π.χ.

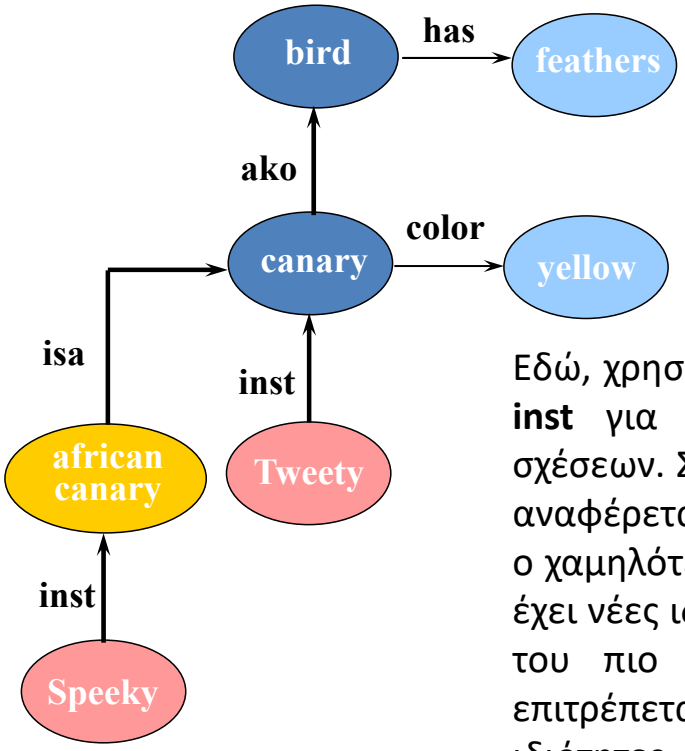
  { Γλυκό(φρούτο) ∨ Ξινό (φρούτο), ← Γλυκό(φρούτο), ← Ξινό (φρούτο)}
- Πώς όμως γνωρίζουμε αν $KB \nvdash f$;
- **«άρνηση ως αποτυχία» (negation as failure-NAF) ← Prolog**
  - Πιο περιορισμένη εκδοχή της CWA
  - δεν μπορεί να ανιχνεύσει καταστάσεις που έχουμε άπειρους κλάδους (infinite branches) στο SLD-δέντρο, παρά μόνο κλάδους που είναι αδιέξοδοι.

Συλλογισμός στην Prolog

- Η SLD-Resolution καθορίζει το ***πώς*** θα σχηματιστεί το δέντρο των αποδείξεων
  - Δεν καθορίζει πώς θα γίνει η διαπέρασή του
- Η Prolog χρησιμοποιεί **depth-first backward chaining**
  - Οι προτάσεις δοκιμάζονται με τη σειρά που είναι γραμμένες στην KB
  - Ο συλλογισμός επομένως είναι **μη πλήρης**, *ακόμα και σε Datalog*
    - *Infinite paths*
    - Επίσης, *redundant paths* λόγω DFS
  - Συμβιβασμός μεταξύ αποδοτικότητας και εκφραστικότητας
- Εξω-λογικά χαρακτηριστικά
  - Αριθμητικές συναρτήσεις: εκτελείται κώδικας και όχι συμπερασμός
    - Π.χ. “X is 4+3”
  - Build-in κατηγορήματα: CUT, ASSERT, RETRACT
    - Δεν αντιστοιχούν στη Λογική και μπορούν να προκαλέσουν συνέπειες ή αλλαγές στην KB

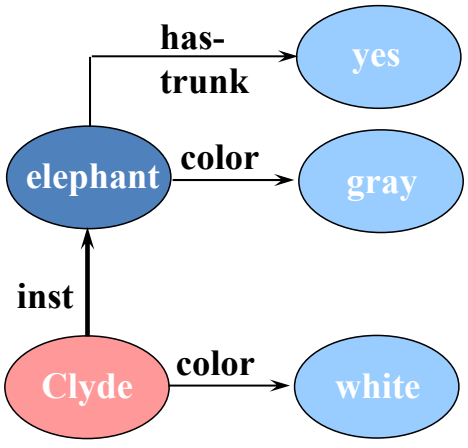
Σημαντικά Δίκτυα – Πλαίσια - Οντολογίες

Ιεραρχικά Δίκτυα με Στιγμιότυπα



Εδώ, χρησιμοποιείται η τριάδα **ako, isa** και **inst** για τη αναπαράσταση ιεραρχικών σχέσεων. Στην περίπτωση αυτή η σχέση isa αναφέρεται σε δύο κόμβους-κλάσεις, όπου ο χαμηλότερα ευρισκόμενος δεν μπορεί να έχει νέες ιδιότητες, μόνο κληρονομεί αυτές του πιο πάνω κόμβου. Το μόνο που επιτρέπεται είναι η αλλαγή τιμών στις ιδιότητες.

Εξ ορισμού (default) Συλλογισμός

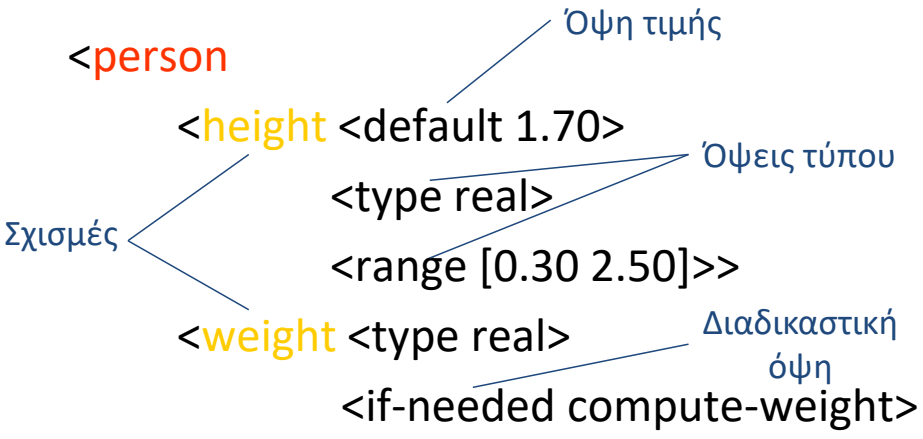


Εδώ, φαίνεται πόσο φυσικά ένα σημαντικό δίκτυο μπορεί να χειριστεί εξαιρέσεις. Ο Clyde είναι λευκός ελέφαντας, ενώ συνήθως οι ελέφαντες είναι γκρίζοι. Όταν ζητήσουμε το χρώμα του Clyde, επειδή συναντάμε πρώτα τον δικό του σύνδεσμο ιδιότητας για το χρώμα, απαντάμε λευκό (white), χωρίς να χρειαστεί να ανέβουμε στον 'elephant'. Δηλ. ένας σύνδεσμος ιδιότητας **επισκιάζει** (ή **καλύπτει**) ένα ίδιο σύνδεσμο που βρίσκεται πιο ψηλά στην ιεραρχία.

Αναπαράσταση με Πλαίσια

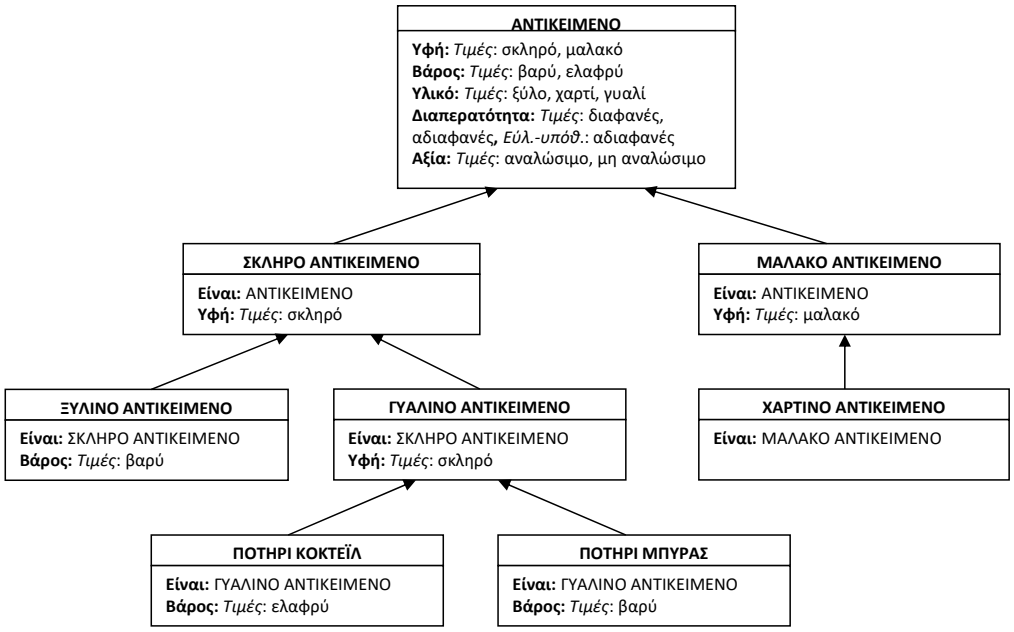
- Τα πλαίσια έχουν:
  - όνομα
  - μία σειρά από σχισμές (slots) που περιγράφουν ιδιότητες
  - όψεις (facets) → τιμές (fillers)
  - δηλωτικές
    - τύπου (type, range)
    - τιμής (value, default)
  - διαδικαστικές → διαδικαστική προσάρτηση
    - (προαιρετικά) προσαρτημένες διαδικασίες (π.χ if-needed, if-added, if-removed)
    - μπορεί να ενεργοποιούνται όταν τα πλαίσια μεταβάλλονται για κάποιο λόγο

Πλαίσια - Παράδειγμα



default ≡ εύλογη-υπόθεση

Ιεραρχίες Πλαισίων - Παράδειγμα (2)



Συλλογισμός με Πλαίσια - Συλλογιστική Απόσταση

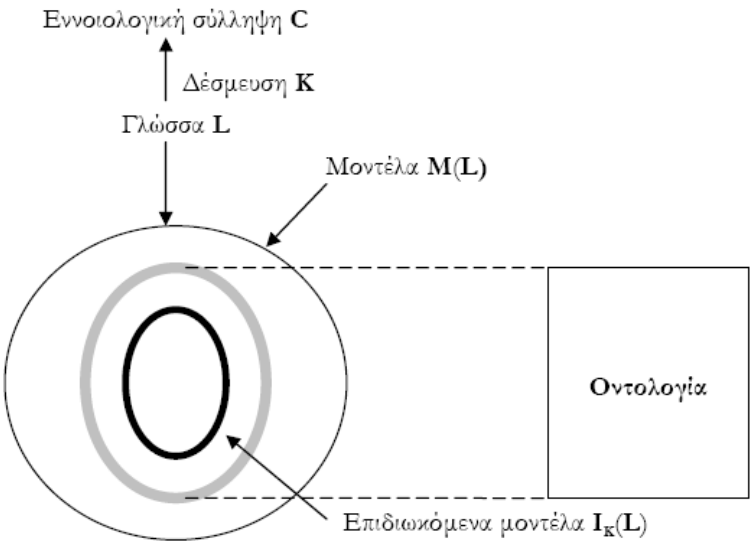
Δεδομένα: Πλαίσιο F, Χαρακτηριστικό S, **Ζητούμενο**: τιμή του S

Αλγόριθμος Βασισμένος στη Συλλογιστική Απόσταση

1. Εφάρμοσε αναζήτηση κατά πλάτος (ή βάθος) ακολουθώντας όλες τις δυνατές διαδρομές από το F προς τα πάνω και αποθήκευσε στη λίστα VALUES όλες τις τιμές που θα βρεις για το S
2. Για κάθε τιμή στη VALUES εξέτασε αν υπάρχει άλλη τιμή που προέρχεται από πλαίσιο που βρίσκεται σε μικρότερη συλλογιστική απόσταση από το F. Αν υπάρχει, διάγραψε την τιμή
3. Αν απομείνουν 0 τιμές, τότε δεν υπάρχει απάντηση. Αν απομείνει μια (1) τιμή είναι η απάντηση. Αν απομείνουν περισσότερες από μία, τότε υπάρχει αντίφαση

Οντολογία κατά Guarino

- **Μοντέλα γλώσσας M:** Δομές που αποδίδουν σημασία στις προτάσεις μιας τυπικής γλώσσας
- **Επιδιωκόμενα μοντέλα:** Τα μοντέλα εκείνα που είναι σύμφωνα με την δέσμευση K
- Η **οντολογία** προσεγγίζει τα επιδιωκόμενα μοντέλα οσοδήποτε κοντά.



Οι οντολογίες ως εννοιολογικές συλλήψεις (conceptualizations) αποτελούν προϊόντα υποκειμενικής κρίσης, οπότε το ίδιο πεδίο ενδιαφέροντος είναι δυνατόν να περιγραφεί με διαφορετικούς τρόπους.

Δομικά στοιχεία οντολογίας

- **Κλάσεις (Classes)**. Οι κλάσεις αναπαριστάνουν έννοιες (concepts) του πεδίου. Μια έννοια μπορεί να είναι οτιδήποτε για το οποίο μπορεί να ειπωθεί κάτι και μπορεί να αφορά μια εργασία, μια λειτουργία, μια ενέργεια, μια ιδέα, κλπ.
- **Θυρίδες (Slots) ή Ιδιότητες (Properties) ή Ρόλοι (Roles)**. Αναπαριστάνουν διάφορα χαρακτηριστικά (features/attributes) των κλάσεων.
- **Όψεις (Facets) ή Περιορισμοί (Restrictions)**. Περιορισμοί που αφορούν τις ιδιότητες/ρόλους.
- **Στιγμιότυπα (Instances)**. Τα στιγμιότυπα αναπαριστούν συγκεκριμένες οντότητες που ανήκουν σε κλάσεις.
- Οι κλάσεις και τα στιγμιότυπα αποτελούν τη **βάση γνώσης** (knowledge base) μιας εφαρμογής.

OWL: Ontology Web Language

- OWL: Γλώσσα Οντολογιών Ιστού
- Επέκταση (μέρους) του RDF(S)
  - Περισσότερη εκφραστικότητα
  - Αντιστοιχεί σε συγκεκριμένη Λογική
  - Δεν έχει τη σημασιολογία του RDF(S)
    - Δεν είναι πλήρως συμβατή με Horn προτάσεις
- Χρήση RDF/XML σύνταξης
  - *Όμως νέο λεξιλόγιο*
    - owl:Class αντί για rdfs:Class
    - owl:ObjectProperty & owl:DataProperty
  - OWL Lite→OWL DL→RDF(S)←→OWL Full

Λογικές Περιγραφής (Description Logics)

- Καλώς ορισμένο υποσύνολο ΛΠΤ
  - Οι τύποι έχουν δύο μεταβλητές
  - Αντιστοιχία με *Σημαντικά Δίκτυα* και *Πλαίσια*
- Παρέχει το λογικό *υπόβαθρο* για διεξαγωγή συλλογισμού στο Σημαντικό Ιστό
  - Η OWL έχει σχεδιαστεί βάσει συγκεκριμένων ΛΠ
- Υπάρχουν υλοποιημένοι αλγόριθμοι και συστήματα
  - RACER, FaCT++, Pellet…
  - Μηχανισμοί *συμπερασμού* (inference engines)

Σύνταξη και Ορολογία

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

- Έννοιες (Concepts)
  - *C, D… αντιστοιχούν στις **κλάσεις***
- Ρόλοι (Roles)
  - *R, P…αντιστοιχούν στις **ιδιότητες***
- Στιγμιότυπα (Instances)
  - *a,o,…αντιστοιχούν στα **άτομα***

Ονοματολογία

ALC	**A**ttribute **L**anguage with **C**omplement
(D)	Απτά Πεδία (τύποι δεδομένων)
F	Συναρτησιακοί Ρόλοι (έχουν το πολύ έναν πληρωτή)
H	Ιεραρχία Ρόλων (έχειΠαιδί, έχειΚόρη)
I	Αντίστροφοι Ρόλοι (έχειΠαιδί, έχειΓονέα)
O	Ονοματικά (Σ/Κ ≡ {Σάββατο} ⊔ {Κυριακή})
Q	Προσδιορισμένοι Περιορισμοί Αριθμού ≥3έχειΠαιδί.Αγόρι
N	Απροσδιόριστοι Περιορισμοί Αριθμού ≤1έχειΠαιδί
R+	Μεταβατικοί Ρόλοι (έχειΑδελφό)
S	ALCR+

Αντιστοιχία με OWL

- Η OWL (DL/Lite) αντιστοιχεί σε ΛΠ
  - OWL Lite → *SHIF*(D)
  - OWL DL → *SHOIN* (D)
  - Οντολογία ≡ Βάση Γνώσης ΛΠ

Αξιώματα

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor

Γεγονότα

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	$\langle$John, Mary$\rangle$: has-child