Ευφυή Συστήματα Υποστήριξης Αποφάσεων

Δημήτρης Αποστόλου

Βιβλιογραφία

- Τεχνητή Νοημοσύνη - Γ' Έκδοση
  - Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου
- Semantic Web Primer
  - Gregoris Antoniou and Frank Van Harmelenn
- Decision Support Systems: A Knowledge Based Approach
  - Clyde Holsapple
- Τεχνητή Νοημοσύνη – Μία σύγχρονη προσέγγιση
  - S. Russell, P. Norvig (Απόδοση στα Ελληνικά: Γ. Ρεφανίδης)

Περιεχόμενα 1ου μέρους

- Συστήματα Αποφάσεων Βασισμένα σε Κανόνες
- Λήψη Αποφάσεων στο Σημασιολογικό Ιστό

Συστήματα Αποφάσεων Βασισμένα σε Κανόνες

Αναπαράσταση με Κανόνες

- Πολύ πρακτικός τρόπος αναπαράστασης για την εξαγωγή συμπερασμάτων
- Αποτελούν τη βάση πολλών συστημάτων γνώσης (knowledge systems)
- Γενικά Πλεονεκτήματα:
  - Κοντά στην ανθρώπινη γνώση
  - Επάρκεια συνεπαγωγών
- Συγκεκριμένα Πλεονεκτήματα:
  - Modularity: Κάθε κανόνας ορίζει ένα μικρό, (σχεδόν) ανεξάρτητο τμήμα της γνώσης
  - Incrementability: Μπορούν να προστεθούν νέοι κανόνες (σχεδόν) ανεξάρτητα από τους υπάρχοντες
  - Modifiability: Οι υπάρχοντες κανόνες μπορούν να αλλάξουν (σχεδόν) ανεξάρτητα από τους υπόλοιπους

Είδη Κανόνων

- Προϋποθέσεις (premises) ή αριστερό μέρος του κανόνα (left hand side - LHS)
  - Συνθήκες (conditions): ακολουθία από κατηγορήματα που συνδέονται με λογικούς τελεστές AND, OR
- Επακόλουθα (consequent) ή δεξιό μέρος του κανόνα (right hand side - RHS)
  - Συμπέρασμα (conclusion): κατηγόρημα
  - Ενέργειες (actions): μία σειρά από εντολές που πρέπει να εκτελεστούν

Μορφές Κανόνων	Εκφράζει	Επεξήγηση
IF *συνθήκες* THEN *συμπέρασμα* *Συνεπαγωγικός (Deductive) κανόνας*	Δηλωτική γνώση	**Αν** οι *συνθήκες* αληθεύουν **τότε** αληθεύει και το *συμπέρασμα*
IF *συνθήκες* THEN *ενέργειες* *Κανόνας Παραγωγής (Production)*	Διαδικαστική γνώση	**Αν** οι *συνθήκες* αληθεύουν **τότε** εκτέλεσε τις *ενέργειες*
ON *συμβάν* IF *συνθήκες* THEN *ενέργειες* *Ενεργός (active) κανόνας*	Διαδικαστική γνώση	**Όταν** συμβεί το *γεγονός (συμβάν)* **Αν** οι *συνθήκες* αληθεύουν **τότε** εκτέλεσε τις *ενέργειες*

Συστήματα Κανόνων

- Συστήματα εξαγωγής συμπερασμάτων (deduction systems)
  - π.χ. Prolog, Datalog
  - Γνώση που δηλώνει μία αλήθεια για τον κόσμο, αλλά δεν αναφέρει ρητά πότε και πώς εφαρμόζεται
- Συστήματα παραγωγής (production systems)
  - π.χ. CLIPS, Flex, Drools
  - Γνώση για το ποιες συγκεκριμένες ενέργειες πρέπει να εκτελεστούν δεδομένης μιας κατάστασης
  - Οι εκτελούμενες ενέργειες επιφέρουν μη-αναστρέψιμα αποτελέσματα
- Ενεργά Συστήματα (re-active systems, active database systems)
  - π.χ. Oracle Triggers, RuleCore

Ενεργοί Κανόνες

- Οι κανόνες παραγωγής δηλώνουν διαδικαστική γνώση
  - Οι κανόνες εκτελούνται "όταν η συνθήκη είναι αληθής"
  - Δεν είναι σαφώς ορισμένο πότε ακριβώς εκτελούνται οι ενέργειες
  - Αν και εκφράζουν διαδικαστική γνώση, η συνθήκη τους περιέχει δηλωτική γνώση.
- Οι ενεργοί κανόνες (active rules) εκφράζουν καθαρά διαδικαστική γνώση
  - Κανόνες οδηγούμενοι από συμβάντα ή γεγονότα (event-driven rules)
- Εκφράζουν με σαφήνεια το πότε ακριβώς ενεργοποιούνται:
  - Όταν συμβεί ένα συγκεκριμένο συμβάν
  - Τότε και μόνο τότε εξετάζεται η συνθήκη τους και αν ικανοποιείται, τότε εκτελούνται οι ενέργειές τους.
- Παραδείγματα συμβάντων:
  - Μία συγκεκριμένη χρονική στιγμή του ρολογιού του συστήματος
  - Ένα πάτημα του αριστερού πλήκτρου του ποντικού ή ενός πλήκτρου του πληκτρολογίου
  - Η επιλογή κάποιου μενού από το χρήστη
  - Η προσπάθεια προσπέλασης ή αλλαγής κάποιων "ευαίσθητων" δεδομένων, κλπ.

Παράδειγμα Αναπαράστασης με Κανόνες

Σύμπτωμα	Πιθανή Βλάβη	Επιδιόρθωση
Ο εκτυπωτής τυπώνει σωστά αλλά τα χρώματα δε τυπώνονται σωστά	Έχει τελειώσει το έγχρωμο μελάνι	Αλλάξτε την κεφαλή με το έγχρωμο μελάνι

- Συνεπαγωγικός Κανόνας
  - IF ο εκτυπωτής τυπώνει σωστά AND
- τα χρώματα δε τυπώνονται σωστά
  - THEN έχει τελειώσει το έγχρωμο μελάνι
- Κανόνας Παραγωγής
  - IF ο εκτυπωτής τυπώνει σωστά AND
- τα χρώματα δε τυπώνονται σωστά
  - THEN αλλάξτε την κεφαλή με το έγχρωμο μελάνι
- Ενεργός Κανόνας
  - ON εκτύπωση
  - IF τα χρώματα δε τυπώνονται σωστά
  - THEN αλλάξτε την κεφαλή με το έγχρωμο μελάνι

Συστήματα Εξαγωγής Συμπερασμάτων

- Βάση κανόνων (rule base): περιέχει ένα σύνολο από κανόνες
- Έλεγχος (control): καθορίζει τον τρόπο εκτέλεσης των κανόνων για την εξαγωγή συμπερασμάτων
  - Ποιοι κανόνες είναι υποψήφιοι για να επιλύσουν το πρόβλημα;
  - Με ποιόν τρόπο θα γίνει η επιλογή;
  - Ποιος από τους κανόνες τελικά θα επιλεγεί;
  - Τι θα γίνει με τους υπόλοιπους κανόνες;

Εξαγωγή Συμπερασμάτων

- Χρησιμοποιείται η συνεπαγωγική συλλογιστική.
  - Υλοποιείται από την ακολουθία εκτέλεσης κανόνων (chaining)
  - Αλγόριθμος που συνδυάζει τα δεδομένα, τους κανόνες και τα ενδιάμεσα συμπεράσματα
- Ορθή ακολουθία εκτέλεσης (forward chaining)
  - Εξετάζονται πρώτα αν οι προϋποθέσεις στο αριστερό μέρος του κανόνα είναι αληθείς έτσι ώστε το συμπέρασμα που αναφέρεται στο δεξιό μέρος να είναι αληθές.
  - Εξετάζονται μόνο οι αληθείς τρόποι απόδειξης
  - Μπορεί να συμπεράνει περισσότερα συμπεράσματα από τα επιθυμητά
  - Ενδείκνυται όταν υπάρχουν λίγα δεδομένα που οδηγούν σε πολλά συμπεράσματα
  - Εφαρμογές: Συστήματα Διάγνωσης, Συστήματα Παραγωγής
- Ανάστροφη ακολουθία εκτέλεσης (backward chaining)
  - Ξεκινά από το δεξιό μέρος του κανόνα και προσπαθεί να βρει αν οι προϋποθέσεις είναι αληθείς
  - Εξετάζονται όλοι οι εναλλακτικοί τρόποι απόδειξης του συμπεράσματος (ακόμα και οι μη-αληθείς) έως ότου αποδειχθεί η αλήθεια του συμπεράσματος (π.χ. Prolog).
  - Ενδείκνυται όταν υπάρχουν λίγα συμπεράσματα και πολλά δεδομένα
  - Το σύστημα ζητά τα δεδομένα με λογική σειρά και μόνο όσα χρειάζονται
  - Εφαρμογές: Συστήματα Ελέγχου Λειτουργίας (Monitoring)

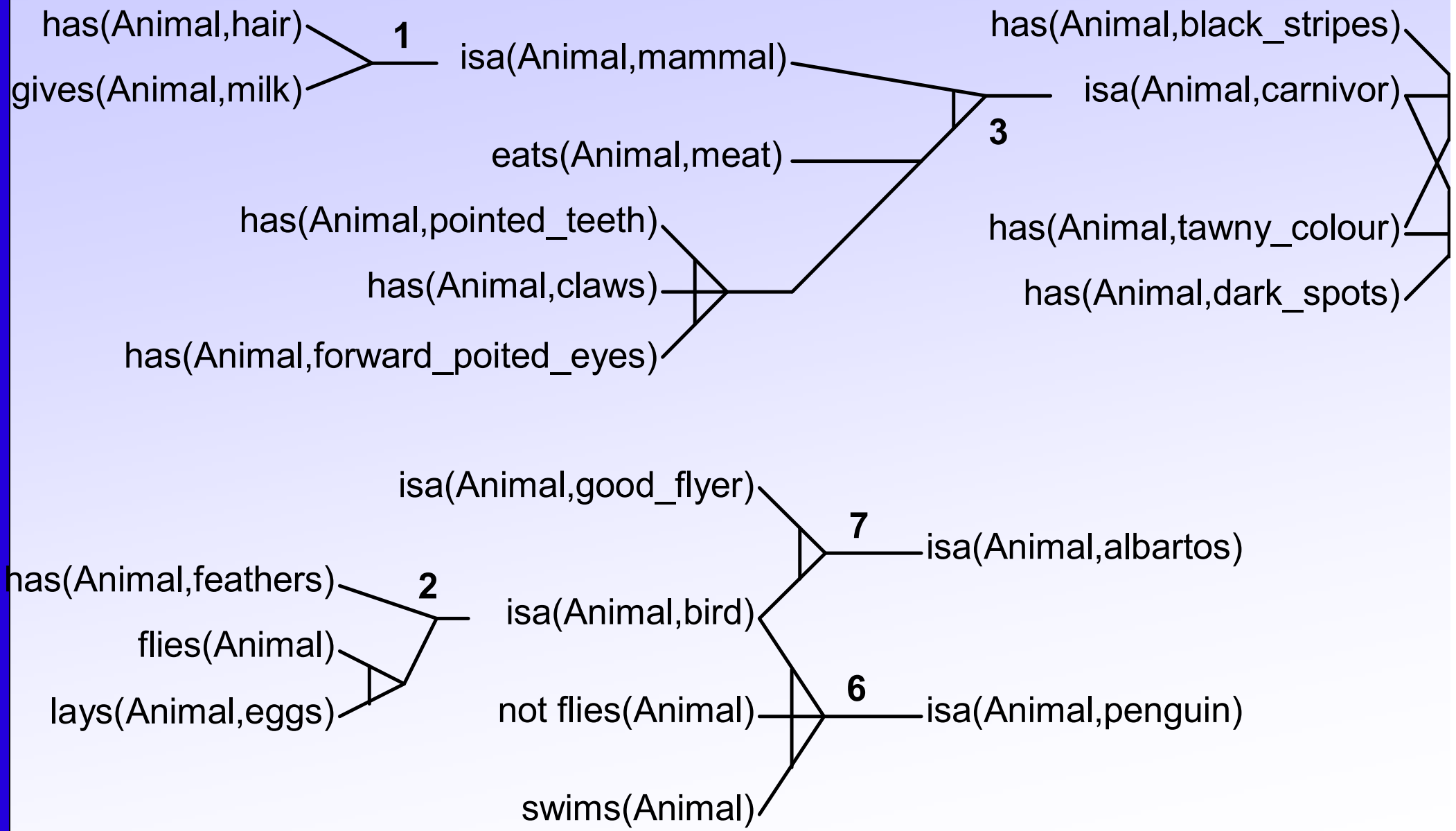
Αναπαράσταση με Κανόνες

Παράδειγμα

```
1:  if has(Animal,hair) or ives(Animal,milk)
  then isa(Animal,mammal).
2:  if  has(Animal,feathers) or
      (flies(Animal) and lays(Animal,eggs))
  then  isa(Animal,bird).
3:  if   isa(Animal,mammal) and
      (eats(Animal,meat) or
      (has(Animal,pointed_teeth) and
      has(Animal,claws) and
      has(Animal,forward_pointing_eyes)))
  then  isa(Animal,carnivore).
4:  if  isa(Animal,carnivore) and
      has(Animal,tawny_colour) and
      has(Animal,dark_spots)
  then  isa(Animal,cheetah).
```

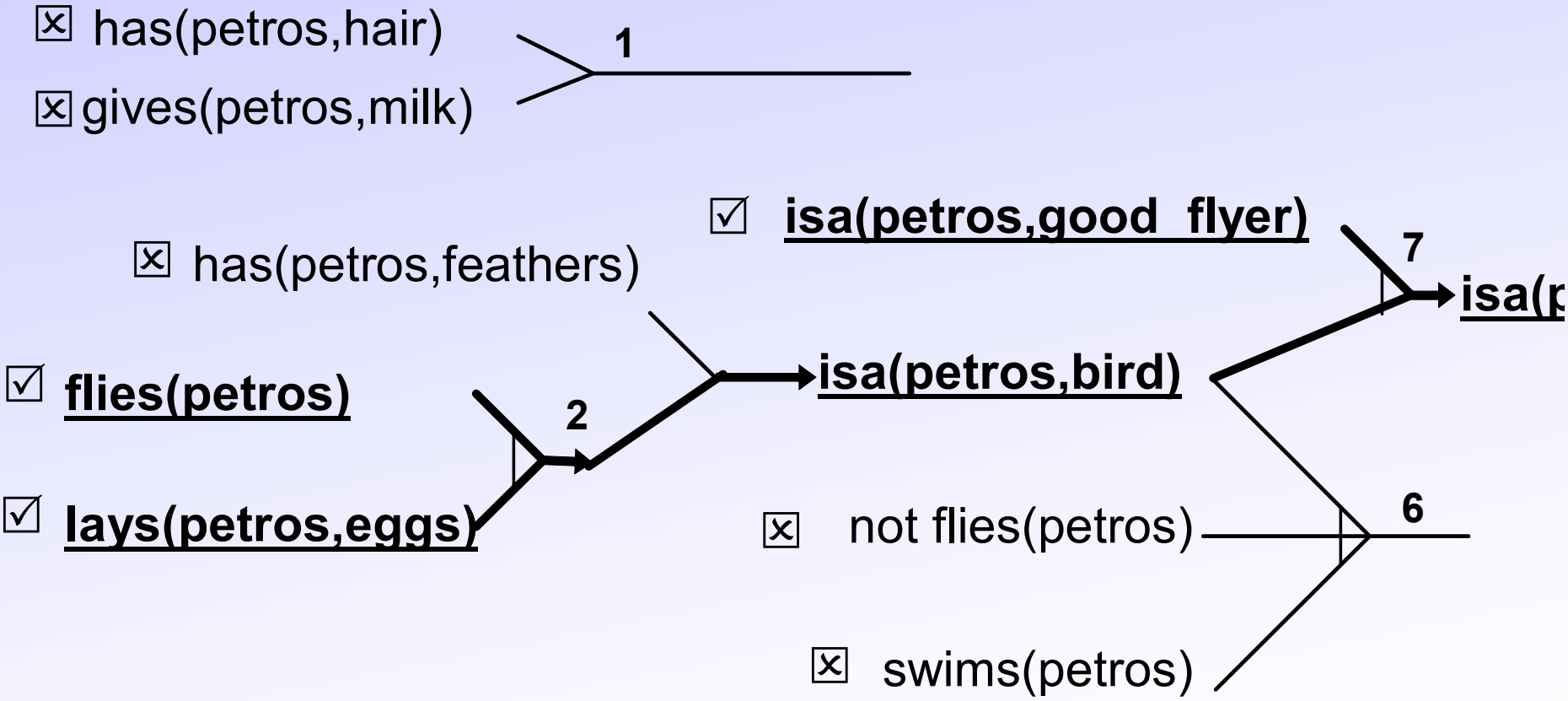
```
5:  if     isa(Animal,carnivore)
and has(Animal,tawny_colour) and
has(Animal,black_stripes)
  then   isa(Animal,tiger).
6:  if  isa(Animaλ,bird) and
        not flies(Animal) and
        swims(Animal)
  then  isa(Animal,penguin).
7:  if  isa(Animal,bird) and
        isa(Animal,good_flyer)
  then   isa(Animal,albatros).
```

Γραφική Αναπαράσταση Κανόνων



Γραφική Αναπαράσταση Εξαγωγής Συμπεράσματος

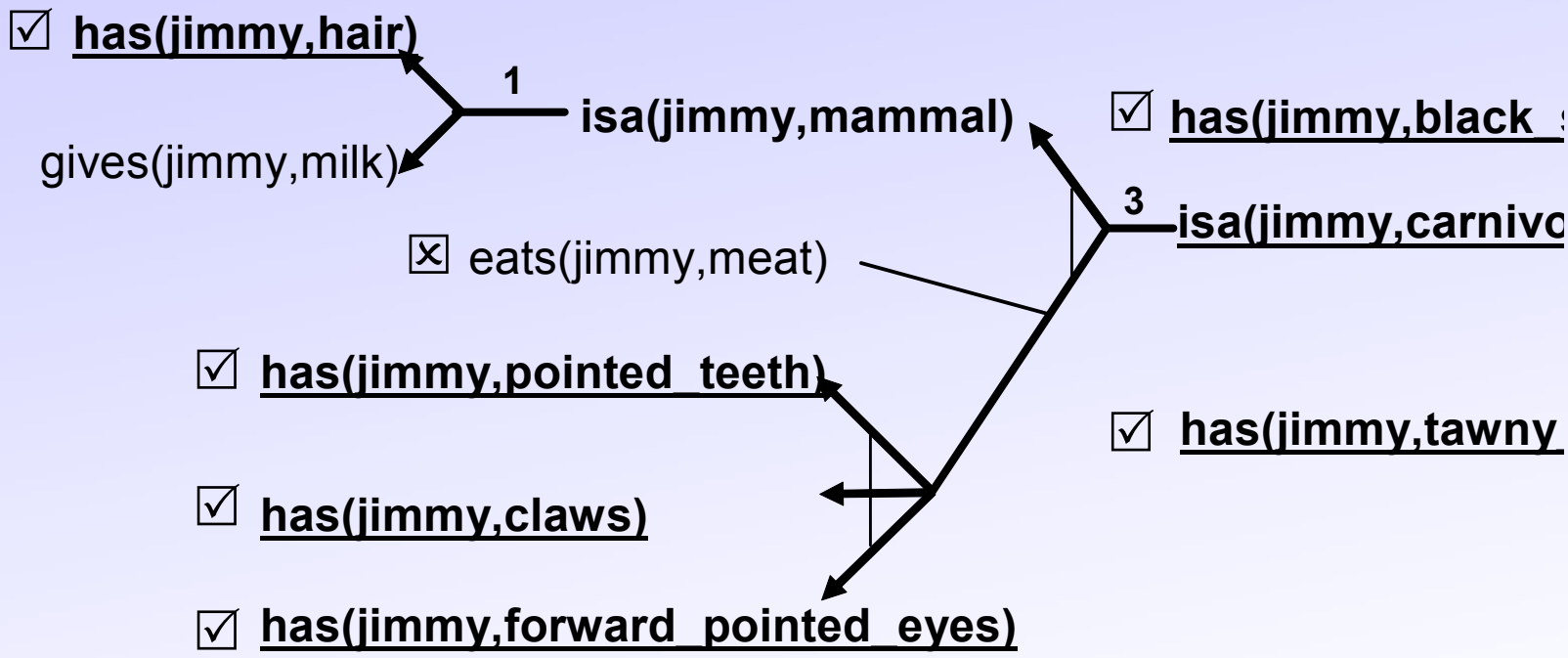
- Ορθή Ακολουθία Εκτέλεσης
- Αρχικά δεδομένα: **flies(petros)**, **lays(petros, eggs)**, **isa(petros, goodflyer)**.



- Παράγονται τα συμπεράσματα: isa(petros, albatros) και isa(petros, bird)

Γραφική Αναπαράσταση Εξαγωγής Συμπεράσματος

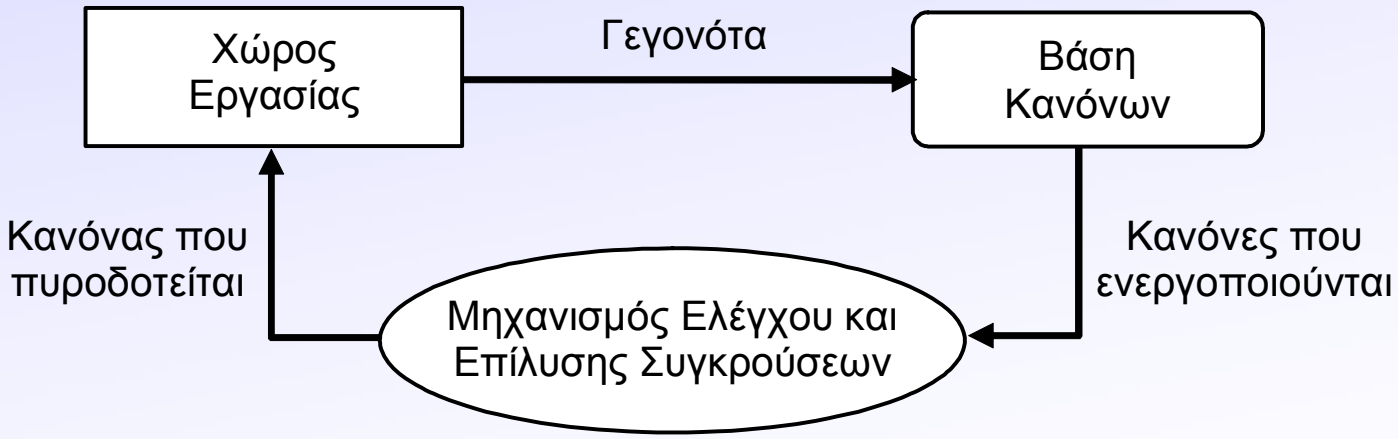
- Ανάστροφη Ακολουθία Εκτέλεσης
  - Ερώτηση: isa(jimmy, tiger)



- Απάντηση: **Yes**

Συστήματα Παραγωγής

- Βάση κανόνων: περιέχει τους κανόνες παραγωγής
- Χώρος εργασίας (working memory): περιέχει γεγονότα
- Αρχικά δεδομένα (data) ή ενδιάμεσα συμπεράσματα (partial conclusions)
- Στοιχεία της μνήμης εργασίας (working memory elements)
- Μηχανισμός ελέγχου (control ή scheduler) και επίλυσης συγκρούσεων (conflict resolution): εκτέλεση των κανόνων βάσει στρατηγικής επίλυσης συγκρούσεων (conflict resolution strategy)



Επίλυση Συγκρούσεων

- Ο κανόνας ενεργοποιείται (triggers) όταν ικανοποιούνται οι συνθήκες του
- Όταν πυροδοτείται (fires) ο κανόνας, τότε εφαρμόζονται/εκτελούνται οι ενέργειές του
- Σύνολο σύγκρουσης (conflict set): Το σύνολο των κανόνων που ενεργοποιούνται
- Ο μηχανισμός ελέγχου καθορίζει ποιος κανόνας θα πυροδοτηθεί
- Στρατηγικές επίλυσης συγκρούσεων:
  - Τυχαία (random).
  - Διάταξης (ordering).
  - Επιλογή του πρόσφατου (recency).
  - Επιλογή του πιο ειδικού (specificity).
  - Αποφυγή επανάληψης (refractoriness).
  - Ανάλυση μέσων-σκοπών (means-ends analysis)
  - Μετα-έλεγχος (μετα-κανόνες)

Στρατηγικές Επίλυσης Συγκρούσεων (1/4)

- Τυχαία (random)
  - Επιλέγεται ένας κανόνας στην τύχη.
  - Συνήθως δεν υπάρχει στα συστήματα κανόνων παραγωγής
  - Αν υπάρχει δεν είναι πραγματικά τυχαία η επιλογή
  - Είναι απροσδιόριστη - εξαρτάται από τις λεπτομέρειες της υλοποίησης
- Διάταξη (ordering)
  - Επιλέγεται ο κανόνας που είναι πρώτος στη σειρά ή έχει μεγαλύτερη προτεραιότητα βάσει κάποιου αριθμητικού μεγέθους
- Αποφυγή επανάληψης (refractoriness)
  - Δεν επιλέγεται ο ίδιος κανόνας με τα ίδια δεδομένα για δεύτερη συνεχόμενη φορά
  - Αποφεύγονται άσκοπες ή ατέρμονες επαναλήψεις
  - Παράδειγμα:
    - Υπάρχουν τα γεγονότα A, B και οι κανόνες:
      - 1: if A then C 2: if B then D
    - Αν εκτελεστεί πρώτα ο 1, μετά θα εκτελεστεί ο 2
    - Ο 1 δε θα εκτελεστεί ξανά, αν και εξακολουθεί να είναι ενεργοποιημένος

Στρατηγικές Επίλυσης Συγκρούσεων (2/4)

- Επιλογή του πρόσφατου (recency)
  - Επιλέγεται ο κανόνας που ενεργοποιείται από τα πιο πρόσφατα δεδομένα που προστέθηκαν στο χώρο εργασίας
  - Ακολουθείται μία χρονικά συνεπής πορεία σκέψης, η οποία είναι επικεντρωμένη και δε διασκορπάται σε διάφορα σημεία
  - Παράδειγμα:
    - Υπάρχουν τα γεγονότα A, B και οι κανόνες:
    - 1: if A then C 2: if B then D 3: if C then E
    - Έστω ότι εκτελείται πρώτα ο 1
    - Μετά θα εκτελεστεί ο 3 (όχι ο 2) γιατί το C είναι πιο πρόσφατο από το B

Στρατηγικές Επίλυσης Συγκρούσεων (3/4)

- Επιλογή του πιο ειδικού (specificity)
  - Επιλέγεται ο κανόνας που είναι πιο ειδικός από τους άλλους
  - Η συνθήκη του εκφράζεται με αναλυτικότερο τρόπο
  - Εξετάζονται πρώτα τα πιο συγκεκριμένα θέματα τα οποία οδηγούν πιθανότατα σε λύση πιο γρήγορα και στη συνέχεια τα πιο γενικά.
  - Παράδειγμα:
    - Υπάρχουν τα γεγονότα A, B, C και οι κανόνες
    - 1: if A and B and C then D 2: if A and B then E
    - Θα εκτελεστεί πρώτα ο 1 γιατί έχει πιο πολλές συνθήκες από τον 2.

Στρατηγικές Επίλυσης Συγκρούσεων (4/4)

- Ανάλυση μέσων-σκοπών (means-ends analysis)
  - Το συνολικό πρόβλημα επιμερίζεται σε απλούστερες διεργασίες (tasks)
  - Κάθε διεργασία υλοποιείται από μία ομάδα κανόνων (cluster)
  - Όταν εκτελείται κάποια διεργασία, τότε οι κανόνες άλλων ομάδων δεν προτιμούνται, παρά μόνο αν δεν υπάρχουν άλλοι ενεργοί κανόνες της ίδιας ομάδας
  - Η αποδεικτική διαδικασία είναι επικεντρωμένη στους τρέχοντες στόχους της.
    - Παράδειγμα:
    - Υπάρχουν τα γεγονότα A, B, C, G1, G2 και οι κανόνες
    - 1: if G1 and A and B and C then D 2: if G2 and A and B then E
    - Τα G1, G2 υποδηλώνουν τη διεργασία που ανήκει ο κανόνας
    - Το G2 είναι πιο πρόσφατο από το G1
    - Θα εκτελεστεί πρώτα ο 2, γιατί ασχολείται με τον πιο τρέχοντα στόχο
    - Ο 1 μπορούσε να έχει προτεραιότητα λόγω άλλης στρατηγικής, (π.χ. specificity)

Μετα-έλεγχος

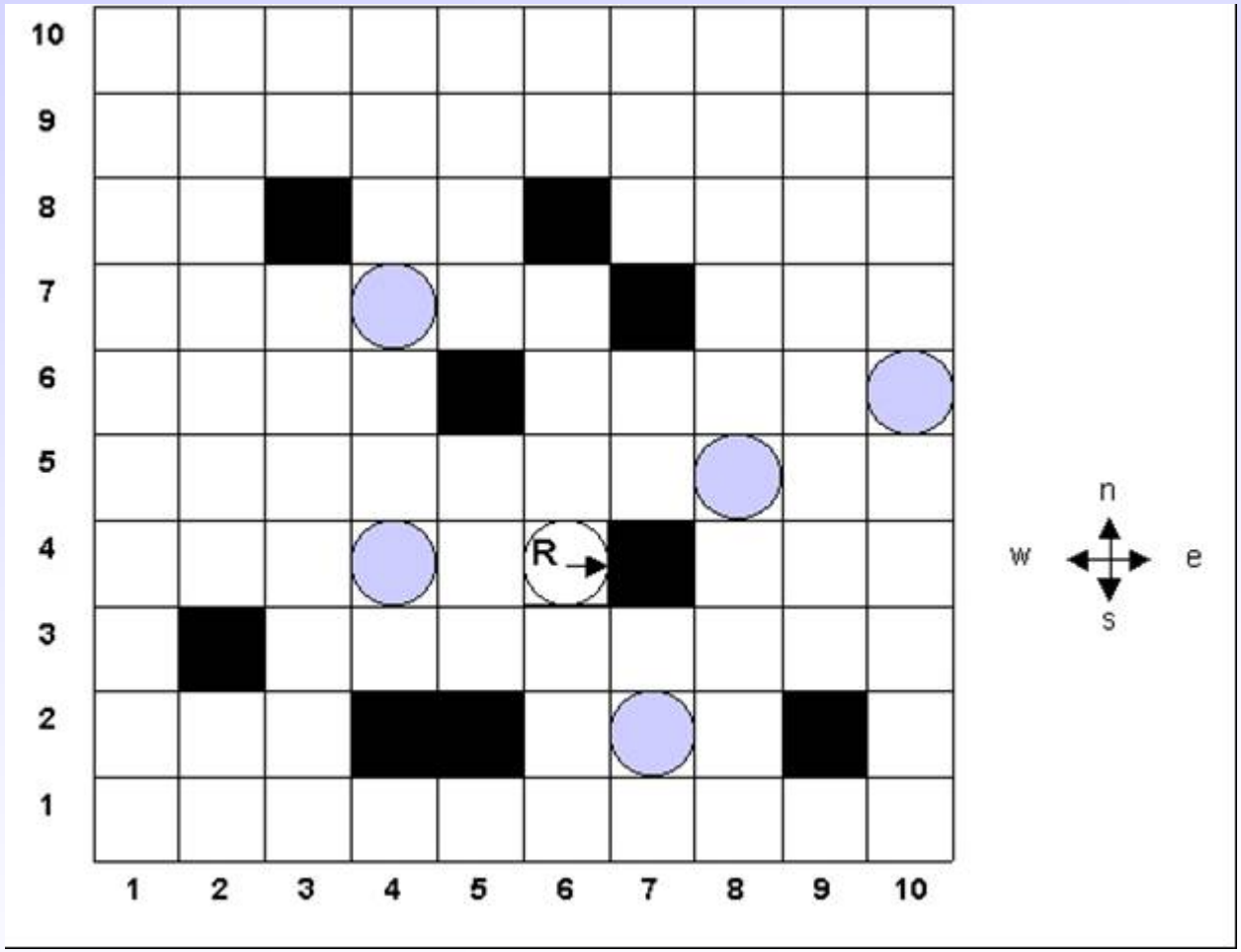
- Τα συστήματα παραγωγής εφαρμόζουν μία ή περισσότερες στρατηγικές επίλυσης συγκρούσεων
- Όταν υπάρχουν πολλές στρατηγικές, πρέπει να υπάρχει προτεραιότητα μεταξύ τους
- Μετα-έλεγχος (meta-control): καθορίζει ποια στρατηγική θα εφαρμοστεί, πού και πότε
- Απλά συστήματα: σταθερή προτεραιότητα
  - Χαμηλότερη τιμή στην τυχαία επιλογή
- Σύνθετα συστήματα: η προτεραιότητα αλλάζει δυναμικά (at run-time)
  - Μετα-κανόνες (meta-rules): κανόνες που καθορίζουν τη σειρά εκτέλεσης άλλων κανόνων

Κύκλος Λειτουργίας Συστήματος Παραγωγής

- Έως ότου δεν μπορεί να εκτελεστεί κανένας κανόνας επανέλαβε:
  - 1. Βρες όλους του κανόνες που οπλίζουν και σχημάτισε το σύνολο συγκρούσεων.
  - 2. Σύμφωνα με το μηχανισμό επίλυσης συγκρούσεων, διάλεξε ένα κανόνα.
  - 3. Πυροδότησε τον κανόνα που διάλεξες στο βήμα 2.
- Στα συστήματα παραγωγής υπάρχει ορθή ακολουθία εκτέλεσης κανόνων
- Εξαγωγή συμπερασμάτων:
  - Δεν έχει νόημα
  - Οι κανόνες παραγωγής αναφέρονται σε ενέργειες που εκτελούνται και όχι σε συμπεράσματα
- Η λειτουργία των κανόνων παραγωγής "παραπέμπει" στη συνεπαγωγική συλλογιστική
  - Υιοθέτηση μιας ειδικής ενέργειας από κάτι που ισχύει γενικά
  - Ταίριασμα των κανόνων που περιέχουν μεταβλητές με δεδομένα στη μνήμη εργασίας που περιέχουν στάθερές

Παράδειγμα Κίνησης Ρομπότ

- Ρομπότ κινείται σε χώρο με εμπόδια
- *Στόχος*: Να αποφύγει τα εμπόδια και όταν βρει κάποιο αντικείμενο, να στείλει ένα μήνυμα και να σταματήσει




```
robot_at(6,4)
direction(e)
choice(w)
choice(s)
choice(n)
choice(e)
obstacle_at(7,4)
obstacle_at(6,8)
obstacle_at(7,7)
. . .
object_at(4,7)
. . .
```

Παρατηρήσεις

- Μνήμη εργασίας:
  - θέση ρομπότ: robot_at(X,Y)
  - κατεύθυνση προς την οποία κινείται: direction(D), D ∈ {e, w, n, s}
  - θέση εμποδίων: obstacle_at(X,Y)
  - θέση αντικειμένων: object_at(X,Y)
  - επιλογή κατεύθυνσης: choice(D), D ∈ {e, w, n, s}
- Ενέργειες κανόνων:
  - addwm: βάλε κάτι στη μνήμη εργασίας
  - delwm: σβήσε κάτι από τη μνήμη εργασίας
  - output: εκτύπωσε ένα μήνυμα στην οθόνη
  - αριθμητικές εκφράσεις.

Κανόνες Κίνησης Ρομπότ

```
1: detect_object: if robot_at(X,Y) and object_at(X,Y)
                  then output('object is found').
2: move_west:     if robot_at(X,Y) and direction(w)
                  then delwm(robot_at(X,Y)) and NX=X-1 and addwm(robot_at(NX,Y)).
3: move_east:     if robot_at(X,Y) and direction(e)
                  then delwm(robot_at(X,Y)) and NX=X+1 and addwm(robot_at(NX,Y)).
4: move_north:    if robot_at(X,Y) and direction(n)
                  then delwm(robot_at(X,Y)) and NY=Y+1 and addwm(robot_at(X,NY)).
5: move_south:    if robot_at(X,Y) and direction(s)
                  then delwm(robot_at(X,Y)) and NY=Y-1 and addwm(robot_at(X,NY)).
6: avoid_obstacle_south:
       if robot_at(X,Y) and NY=Y-1 and obstacle_at(X,NY) and direction(s) and choice(ND)
       then delwm(direction(s)) and addwm(direction(ND)).
7: avoid_obstacle_west:
       if robot_at(X,Y) and NX=X-1 and obstacle_at(NX,Y) and direction(w) and choice(ND)
       then delwm(direction(w)) and addwm(direction(ND)).
8: avoid_obstacle_north:
       if robot_at(X,Y) and NY=Y+1 and obstacle_at(X,NY) and direction(n) and choice(ND)
       then delwm(direction(n)) and addwm(direction(ND)).
9: avoid_obstacle_east:
       if robot_at(X,Y) and NX=X+1 and obstacle_at(NX,Y) and direction(e) and choice(ND)
            then delwm(direction(e)) and addwm(direction(ND)).
```

MOVE

Change Direction

Στρατηγική Επίλυσης Κίνησης Ρομπότ

- Αποφυγή Επανάληψης (ΑΕ)
  - Βοηθά να μην κολλήσει το ρομπότ σε εμπόδιο, επιλέγοντας συνεχώς την κατεύθυνση προς την οποία βρίσκεται το εμπόδιο
- Επιλογή του πιο Ειδικού (ΕΕ)
  - Δίνει προτεραιότητα στην αποφυγή εμποδίων (κανόνες 6 - 9)
- Τυχαία Επιλογή (ΤΕ)
  - Το ρομπότ επιλέγει τυχαία μία από τις 4 κατευθύνσεις

Παρακολούθηση Εκτέλεσης (1/3)

Κύκλος	Μνήμη Εργασίας	Σύνολο Συγκρούσεων	Στρατηγική	Κανόνας που πυροδοτεί
1	robot_at(6,4) direction(e) choice(w) choice(n) choice(s) choice(e) obstacle_at(7,4) obstacle_at(6,8) . . . object_at(4,7) . . .	{3, 9 (ND=w), 9 (ND=n), 9 (ND=s), 9 (ND=e)}	EE TE	9:avoid_obstacle_east (ND=n)
2	robot_at(6,4) direction(n) . . .	{4}	-	4: move_north
3	robot_at(6,5) direction(n) . . .	{4}	-	4: move_north

Επιλέγεται ένας (9), με choice(n)

Κανόνες που οπλίζουν απο τα δεδομένα της Μνήμης Εργασίας.
Ο (3) από το direction(e) και ο (9) από το direction(e) και από το obstacle(7,4)

Παρακολούθηση Εκτέλεσης (2/3)

Κύκλος	Μνήμη Εργασίας	Σύνολο Συγκρούσεων	Στρατηγική	Κανόνας που πυροδοτεί
4	robot_at(6,6) direction(n) . . .	{4}	-	4: move_north
5	robot_at(6,7) direction(n) . . . obstacle_at(6,8) . . .	{4, 8 (ND=w), 8 (ND=n), 8 (ND=s), 8 (ND=e)}	EE TE	8:avoid_obstacle_north (ND=n)
6	robot_at(6,7) direction(n) . . . obstacle_at(6,8) . . .	{4, 8 (ND=w), 8 (ND=n), 8 (ND=s), 8 (ND=e)}	AE EE TE	8:avoid_obstacle_north (ND=e)
7	robot_at(6,7) direction(e) . . . obstacle_at(7,7) . . .	{3, 9 (ND=w), 9 (ND=n), 9 (ND=s), 9 (ND=e)}	EE TE	9: avoid_obstacle_east (ND=w)

Παρακολούθηση Εκτέλεσης (3/3)

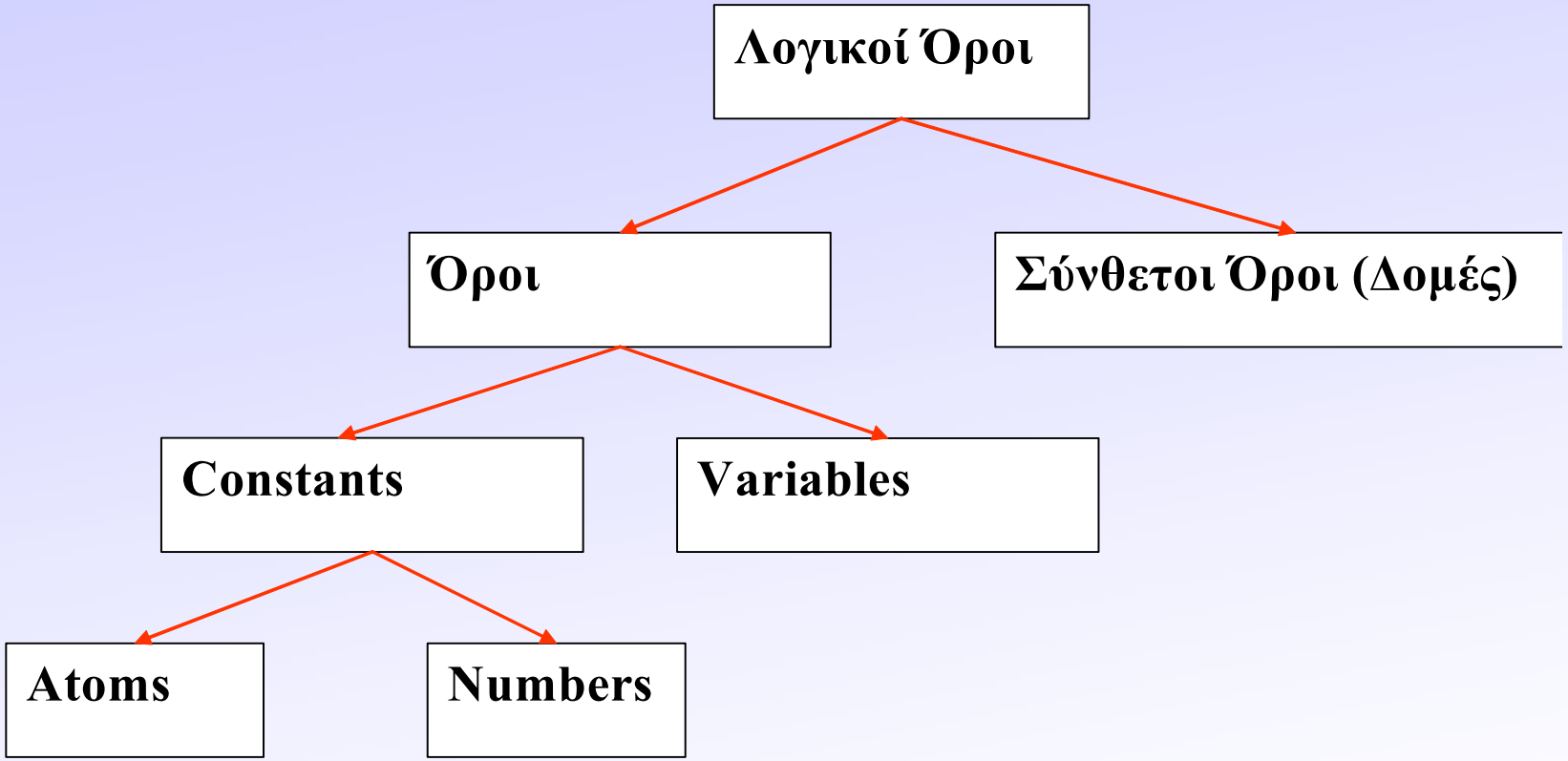
Κύκλος	Μνήμη Εργασίας	Σύνολο Συγκρούσεων	Στρατηγική	Κανόνας που πυροδοτεί
8	robot_at(6,7) direction(w) . . .	{2}	-	2: move_west
9	robot_at(5,7) direction(w) . . .	{2}	-	2: move_west
10	robot_at(4,7) direction(w) object_at(4,7) . . .	{1,2}	EE TE	1: detect_object

Λογικός Προγραμματισμός

Prolog

- Υπάρχουν τρεις βασικές δηλώσεις:
  - τα γεγονότα (facts),
  - οι ερωτήσεις (queries) και
  - οι κανόνες (rules).
- Υπάρχει μια μοναδική δομή δεδομένων: ο λογικός όρος (logical term) ή απλά όρος.

Δομές Δεδομένων στην Prolog



Γεγονότα

- Το πιο απλό είδος της δήλωσης ονομάζεται **γεγονός (fact)**.
- Τα γεγονότα είναι ένα μέσο καθορισμού μιας σχέσης που ισχύει ανάμεσα στα αντικείμενα. Ένα παράδειγμα είναι:

  father(abraham,isaac).

- Αυτό το γεγονός λέει ότι ο *Abraham* είναι ο πατέρας του *Isaac*, ή ότι ο συσχετισμός *πατέρας (father)* ισχύει ανάμεσα στις μονάδες, οι οποίες ονομάζονται *abraham* και *isaac*.
- Άλλο ένα όνομα για μια σχέση είναι το **κατηγόρημα (predicate)**.
- Τα ονόματα των μονάδων είναι γνωστά ως **άτομα (atoms)**.
- Το κατηγόρημα father έχει 2 ορίσματα. Δηλ. ***father/2.***

Ερωτήσεις

- Η δεύτερη μορφή μιας δήλωσης στον λογικό προγραμματισμό είναι η ερώτηση (query).
- Οι ερωτήσεις είναι το μέσο απόδοσης πληροφοριών από ένα λογικό πρόγραμμα.
- Με μια ερώτηση, ρωτάμε αν ισχύει κάποιος συσχετισμός ανάμεσα σε αντικείμενα.
- Για παράδειγμα η ερώτηση

  father(abraham,isaac)?
- ρωτάει, εάν η σχέση πατέρας (father) ισχύει ανάμεσα στον abraham και στον isaac.
- Συζευκτική ερώτηση είναι μια σύζευξη στόχων που εκφράζονται σε μια ερώτηση, για παράδειγμα

  father(terach,X), father(X,Y)? ή γενικά, Q1, . . ., Qn?.

Κανόνες

- Αυτό μας φέρνει στην τρίτη και πιο σημαντική δήλωση στον λογικό προγραμματισμό, ο κανόνας (rule), ο οποίος μας καθιστά ικανούς στο να καθορίσουμε νέες σχέσεις στους όρους των υφιστάμενων σχέσεων.
- Ο X είναι εγγονός του Y αν ο X είναι παιδί κάποιου Z και ο/η Z είναι παιδί του Y.

```
grandchild (X,Y) :- child (X,Z), child (Z,Y).
```

head (κεφαλή) body (σώμα)

Κανόνες και Συζεύξεις

- A man is happy if he is rich and famous might translate to:

```
happy(Person):-
        man(Person),
        rich(Person),
        famous(Person).
```

- The “,” indicates the conjunction and is roughly equivalent to the ^ of predicate calculus. Therefore, read “,” as “and”.
- In this single clause, the logical variable Person refers to the same object throughout.
- By the way, we might have chosen any name for the logical variable other than Person.
  - It is common practice to name a logical variable in some way that reminds you of what kind of entity is being handled.

Κανόνες και Διαζεύξεις

- Someone is happy if they are healthy, wealthy or wise, translates to:

```
happy(Person):-
                    healthy(Person).
happy(Person):-
                    wealthy(Person).
happy(Person):-
                    wise(Person).
```

- Note how we have had to rewrite the original informal statement into something like:
  - Someone is happy if they are healthy OR
  - Someone is happy if they are wealthy OR
  - Someone is happy if they are wise

Παράδειγμα

```
related (X,X).
related (X,Y) :- married (X,Y).
related (X,Y) :- child (X,Z), related (Z,Y).
related (X,Y) :- child (Z,X), related (Z,Y).
```

Λογικές Μεταβλητές και Κανόνες

- The scope rule for Prolog is that two uses of an identical name for a logical variable only refer to the same object if the uses are within a single clause.
- Therefore in:

  happy(X):- healthy(X).

  wise(X):- old(X).
- the two references to X in the first clause do not refer to the same object as the references to X in the second clause.
- Do not assume that the word logical is redundant. It is used to distinguish between the nature of the variable as used in predicate calculus and the variable used in imperative languages like BASIC, FORTRAN and so on.
- In those languages, a variable name indicates a storage location which may `contain' different values at different moments in the execution of the program.
- The logical variable cannot be overwritten with a new value.

Στρατηγική Αναζήτησης στην Prolog

- «Κατά βάθος»
- Program Database

```
woman(jean).
man(fred).
woman(jane).
woman(joan).
woman(pat).

?- woman(jane).
```

- Prolog searches through the set of clauses from top to bottom. First, Prolog examines woman(jean) and finds that woman(jane) does not match.
- Also, it is obvious that the next clause man(fred). doesn't match either.
- Prolog then comes to look at the third clause and it finds what we want.

Παράδειγμα

- Program database

```
woman(jean).
man(fred).
wealthy(fred).
happy(Person):-
                    woman(Person),
                    wealthy(Person).

?- happy(jean).
no
```

- Why? Prolog tries to match:

```
happy(jean)
  against
happy(Person)
```

- We call this matching process unification. What happens here is that the logical variable Person gets bound to the atom jean. You could paraphrase “bound" as “is temporarily identified with".
- To solve our problem, Prolog must set up two subgoals. But we must make sure that, since Person is a logical variable, that everywhere in the rule that Person occurs we will replace Person by jean.
- We now have something equivalent to:

  happy(jean):- woman(jean), wealthy(jean).
- So the two subgoals are:

  woman(jean)

  wealthy(jean)

- Here we come to our next problem. In which order should Prolog try to solve these subgoals?
  - The answer is that the standard way to choose the subgoal to work on first is again based on the way we read (in the west)! We try to solve the subgoal woman(jean) and then the subgoal wealthy(jean). There is only one possible match for woman(jean): our subgoal is successful.
- However, we are not finished until we can find out if wealthy(jean).
- There is a possible match but we cannot unify
- wealthy(fred)
- with
- wealthy(jean)
- (However, Prolog is not finished yet. Once we reach wealthy(Person) with Person/jean and it fails we move back (backtracking) to the goal woman(Person) and break the binding for Person (because this is where we made the binding Person/jean). We now start going from left to right again.)

Κανόνες και Αβεβαιότητα

Συντελεστές Βεβαιότητας (Certainty Factors) (1/2)

- Αριθμητικές τιμές που εκφράζουν τη βεβαιότητα για την αλήθεια μιας πρότασης ή γεγονότος. Πρωτο-εισήχθησαν στο έμπειρο σύστημα MYCIN

  if γεγονός then υποθετικό συμπέρασμα με βεβαιότητα CF
- Παράδειγμα: if πυρετός then γρίπη CF 0.8
- Παίρνουν τιμές στο διάστημα [-1, +1]

  –1 : απόλυτη βεβαιότητα για το ψευδές της πρότασης.
  +1 : απόλυτη βεβαιότητα για την αλήθεια της πρότασης.
  0 : άγνοια.
- Τιμές βεβαιότητας και στην τιμή του γεγονότος του κανόνα:
- Παράδειγμα: if πυρετός CF1 0.7 then γρίπη CF 0.8
  - τελική βεβαιότητα κανόνα: 0.7x0.8=0.56
- Αν υπάρχουν περισσότερα από ένα γεγονότα στο αριστερό τμήμα του κανόνα τα οποία συνδέονται με AND (ή OR) τότε ως συντελεστής βεβαιότητας του αριστερού τμήματος θεωρείται η μικρότερη (ή η μεγαλύτερη) τιμή CF που εμφανίζεται

Συντελεστές Βεβαιότητας (Certainty Factors) (2/2)

- Αν δύο διαφορετικοί κανόνες συνάγουν το ίδιο υποθετικό συμπέρασμα με βεβαιότητες CFp και CFn, τότε η συνολική βεβαιότητα είναι:
- Αν CFp και CFn > 0, τότε: CF = CFp+CFn•(1-CFp) = CFp+CFn–CFn•CFp
- Αν CFp και CFn < 0, τότε: CF = CFp+CFn• (1+CFp) = CFp+CFn+CFn•CFp
- Αν CFp και CFn ετερόσημα, τότε:

$$CF = \frac{CF_p + CF_n}{1 - \min(|CF_p|, |CF_n|)}$$

- Παράδειγμα: if πυρετός then γρίπη CF 0.8
  if βήχας then γρίπη CF 0.5
- **Συμπερασματικά**:
- Αντί για συχνότητες εμφάνισης γεγονότων που πρέπει να μετρηθούν, χρησιμοποιούνται συντελεστές βεβαιότητας που έχουν εκτιμηθεί από ειδικούς.
- Οι υπολογισμοί κατά το συνδυασμό βεβαιοτήτων είναι απλούστεροι, λόγω της παραδοχής της ανεξαρτησίας των γεγονότων.
- Πρέπει να αποφεύγεται η ταυτόχρονη χρήση κανόνων που αναστρέφουν τη σχέση αιτίας-αποτελέσματος. Π.χ. *if A then B* και *if B then A*

Παράδειγμα

- Έστω ότι δύο κανόνες οδηγούν στο ίδιο υποθετικό συμπέρασμα Β, κάτω όμως από διαφορετικές παραδοχές, δηλαδή:

  if A then B CF 0.8

  if C AND D AND E then B CF 0.6

- Αν ο χρήστης εισάγει τα δεδομένα A, C, D και E με βεβαιότητες:

  CF(A)=0.5, CF(C)=0.9, CF(D)=0.7 και CF(E)=0.5 τότε:

- Η ενεργοποίηση του πρώτου κανόνα δίνει: CFp(B)=0.5 * 0.8 =0.4
- Η ενεργοποίηση του δεύτερου κανόνα δίνει:

  CFn(B)=0.6 * min(0.9, 0.7, 0.5) = 0.6 * 0.5 = 0.3

- Επειδή τα CFp και CFn είναι και τα δύο θετικά, η συνολική βεβαιότητα του υποθετικού συμπεράσματος Β θα είναι:
  CF(B) = 0.4 + 0.3 - (0.4 x 0.3) = 0.58

Προσέγγιση Dempster-Shafer (D-S) (1/2)

- Δεν απαιτείται η συλλογή όλων των απλών και των υπό συνθήκη πιθανοτήτων.
- Λογισμός με αριθμητικές τιμές *πεποίθησης* (*belief*)
- Αν U={A, B, C} πιθανές ασθένειες τότε Pow είναι το σύνολο των υποσυνόλων του U:
  Pow(U) = { {}, {A}, {B}, {C}, {A, B}, {A, C}, {B, C}, {A, B, C} }
  πιθανές διαγνώσεις.
- Διαζευγμένες Προτάσεις: {A, B} σημαίνει "ασθένεια A ή B".
- Στοιχεία του U που δεν ανήκουν σε ένα στοιχείο του Pow(U), (π.χ. η ασθένεια C στο {A, B}), κάνουν σαφή την άρνηση του αντίστοιχου υποθετικού συμπεράσματος.
- {}: *null hypothesis*

Προσέγγιση Dempster-Shafer (2/2)

- Η *βασική κατανομή πιθανότητας* (*basic probability assignment - bpa*) είναι μία απεικόνιση:

  m: Pow(U) $\rightarrow$ [0,1]

- η οποία αναθέτει μία τιμή πεποίθησης για κάθε στοιχείο του Pow(U)
- Είναι δηλαδή το μέτρο της πεποίθησης που υπάρχει για το κατά πόσο ισχύει το υποθετικό συμπέρασμα που εκφράζεται με το συγκεκριμένο στοιχείο του U.
- *η πεποίθηση m({A, B})=0.3, δε μοιράζεται στα {A} και {B} αλλά αφορά το {A, B}.*
- *ισχύει m({})=0*

$$\sum_{X \in Pow(U)} m(X) = 1$$

- Επειδή θεωρούμε οτι το υποθετικό συμπέρασμα βρίσκεται κάπου μέσα στα στοιχεία του Pow(U)
- Η συνολική πεποίθηση (*belief*) ότι ένα στοιχείο του U ανήκει στο X καθώς και στα τυχόν υποσύνολα του X συμβολίζεται με *Bel(X)*

$$Bel(X) = \sum_{Y \subseteq X} m(Y)$$

Κανόνας Dempster-Shafer

- Αν *m1* και *m2* δύο ανεξάρτητες εκτιμήσεις (βασικές κατανομές πιθανότητας) που αποδίδουν κάποιο βαθμό πεποίθησης στα στοιχεία του Pow(U), τότε αυτές συνδυάζονται σε μία τρίτη εκτίμηση *m3=m1⊕m2* με τρόπο που ορίζεται με τον *κανόνα D-S*:

$$m_3(\mathrm{A}) = m_1 \oplus m_2(\mathrm{A}) = \frac{\sum_{\mathrm{X,Y}\in\mathrm{U}:\mathrm{X}\cap\mathrm{Y}=\mathrm{A}} m_1(\mathrm{X}) \bullet m_2(\mathrm{Y})}{1-\sum_{\mathrm{X,Y}\in\mathrm{U}:\mathrm{X}\cap\mathrm{Y}=\varnothing} m_1(\mathrm{X}) \bullet m_2(\mathrm{Y})}$$

$$A\in Pow(U)$$

Παράδειγμα: Διάγνωση Ασθένειας

- Έστω $U=\{A, B, C\}$ το σύνολο των δυνατών ασθενειών που μπορεί να διαγνωσθούν.
- Πιθανές Διαγνώσεις $Pow(U)=\{\{\}, \{A\}, \{B\}, \{C\}, \{A,B\}, \{A,C\}, \{B,C\}, \{A,B,C\}\}$
- $m(\ \{\{\ \}, \{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}, \{B, C\}, \{A, B, C\}\ \}\ ) = 1$
- υποδηλώνει τη βεβαιότητα ότι η διάγνωση βρίσκεται κάπου στα στοιχεία του $Pow(U)$ αλλά ελλείψει άλλων ενδείξεων δεν είναι δυνατό να δοθεί ιδιαίτερη βαρύτητα σε κάποιο
- Έστω ότι γίνεται διαθέσιμη επιπλέον πληροφορία, (π.χ. πραγματοποιούνται ιατρικές εξετάσεις) και προκύπτει ότι η ασθένεια είναι μία από τις A ή B με βαθμό πίστης 0.7
- $m1(\ \{A, B\}\ ) = 0.7$
- $m1(\ \{\{\ \}, \{A\}, \{B\}, \{C\}, \{A, C\}, \{B, C\}, \{A, B, C\}\}\ ) = 0.3$
- Δηλαδή, η έλλειψη πίστης σε ένα από τα υποθετικά συμπεράσματα του $Pow(U)$, ισοδυναμεί αυτόματα με ισόποσο βαθμό πίστης στα υπόλοιπα στοιχεία του $Pow(U)$, χωρίς όμως να δίνεται ιδιαίτερη προτίμηση σε κάποιο από αυτά.
- Πώς μπορεί να συνδυαστούν δύο ανεξάρτητες εκτιμήσεις (π.χ. δύο ιατρών) σε μία;

Παράδειγμα: Συνδυασμός Διαγνώσεων

- Έστω ότι δύο γιατροί εξετάζουν ανεξάρτητα τον ασθενή και δίνουν την εκτίμησή τους *m1* και *m2* αντίστοιχα, για την αρρώστια από την οποία αυτός πάσχει.

	Γιατρός 1		Γιατρός 2	
Δυνατές περιπτώσεις διάγνωσης	m_1	Bel_1	m_2	Bel_2
{A}	0.05	0.05	0.15	0.15
{B}	0	0	0	0
{C}	0.05	0.05	0.05	0.05
{A, B}	0.15	0.2	0.05	0.2
{A, C}	0.1	0.2	0.2	0.4
{B, C}	0.05	0.1	0.05	0.1
{A, B, C}	0.6	1	0.5	1

$$m_3(A) = m_1 \oplus m_2(A) = \frac{\sum_{X,Y \in Pow(U): X \cap Y = A} m_1(X) \cdot m_2(Y)}{1 - \sum_{X,Y \in Pow(U): X \cap Y = \emptyset} m_1(X) \cdot m_2(Y)}$$
$$A \in Pow(U)$$

$$Bel(X) = \sum_{Y \subseteq X} m(Y)$$

- π.χ. *Bel1*({*A*,*B*}) = *m1*({*A*,*B*}) + *m1*({*A*}) + *m1*({*B*}) = 0.015 + 0.05 + 0 = 0.2

Παράδειγμα: Συνδυασμός Εκτιμήσεων (1/2)

$m_3=m_1 \oplus m_2$		m_1						
		{A}	{B}	{C}	{A,B}	{A,C}	{B,C}	{A,B,C}
m_2		0.05	0	0.05	0.15	0.1	0.05	0.6
{A}	0.15	{A} .0075	{ } 0	{ } .0075	{A} .0225	{A} .015	{ } .0075	{A} .09
{B}	0	{ } .0	{B} 0	{ } .0	{B} .0	{ } .0	{B} .0	{B} .0
{C}	0.05	{ } .0025	{ } 0	{C} .0025	{ } .0075	{C} .005	{C} .0025	{C} .03
{A,B}	0.05	{A} .0025	{B} 0	{ } .0025	{A,B} .0075	{A} .005	{B} .0025	{A,B} .03
{A,C}	0.2	{A} .01	{ } 0	{C} .01	{A} .03	{A,C} .02	{C} .01	{A,C} .012
{B,C}	0.05	{ } .0025	{B} 0	{C} .0025	{B} .0075	{C} .005	{B,C} .0025	{B,C} .03
{A,B,C}	0.5	{A} .025	{B} 0	{C} .025	{A,B} .075	{A,C} .05	{B,C} .025	{A,B,C} .3

Παράδειγμα: Συνδυασμός Εκτιμήσεων (2/2)

Δυνατές περιπτώσεις διάγνωσης	m_3	Bel_3
{A}	0.21	0.21
{B}	0.01	0.01
{C}	0.09	0.09
{A, B}	0.12	0.34
{A, C}	0.20	0.50
{B, C}	0.06	0.16
{A, B, C}	0.31	1.00

$$Bel(X) = \sum_{Y \subseteq X} m(Y)$$

- Η αρχική εκτίμηση ότι η ασθένεια είναι μία από τις Α ή Β αποδυναμώθηκε.
- η διάγνωση βρίσκεται μάλλον στο σύνολο **{A,C}**
- επειδή **Bel3({A})>Bel3({C})**, αρχίζει να διαφαίνεται ότι η τελική διάγνωση είναι η Α
- Η παραπάνω συνδυασμένη εκτίμηση μπορεί να συνδυαστεί εκ νέου με μια άλλη εκτίμηση (π.χ. 3ου ιατρού).

Κανόνες και Ασάφεια

Ασάφεια (Fuzziness)

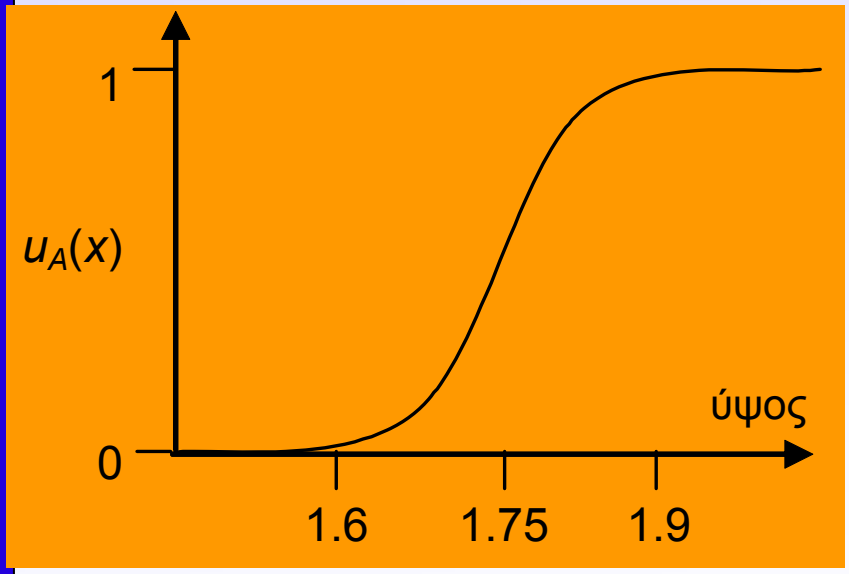
- Έννοια που σχετίζεται με την ποσοτικοποίηση της πληροφορίας και οφείλεται κυρίως σε *μη-ακριβή* (*imprecise*) δεδομένα.
- *"Ο Νίκος είναι ψηλός"*
- Το πρόβλημα οφείλεται στην αντίληψη που έχει ο καθένας για λεκτικούς προσδιορισμούς ποσοτικών μεγεθών (*σημασιολογική ασάφεια*)
- Εγγενές χαρακτηριστικό της γλώσσας.
- *Ασαφής Λογική* (*fuzzy logic*): υπερσύνολο της κλασικής λογικής
- χειρίζεται τιμές αληθείας μεταξύ του "απολύτως αληθούς" και του "απολύτως ψευδού".
- *Θεωρία Ασαφών Συνόλων* (*Fuzzy Set Theory*) - Lofti Zadeh '60

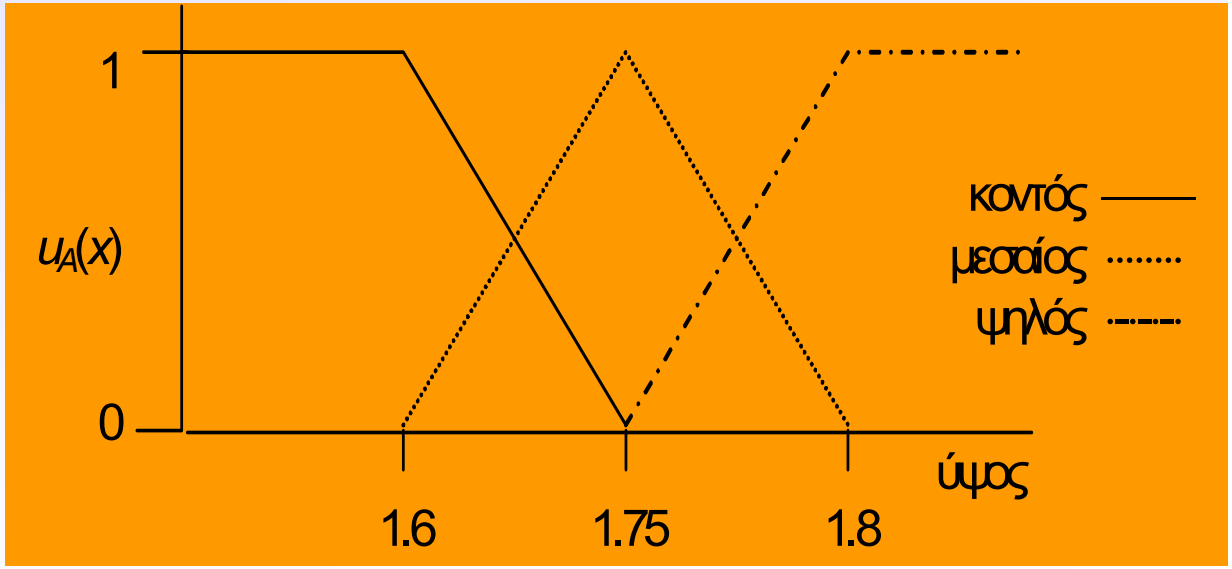
Βασικές Έννοιες Ασαφών Συνόλων

- *Ασαφές Σύνολο* (*fuzzy set*) A: ένα σύνολο διατεταγμένων ζευγών (x, uA(x)) όπου $x \in X$ και $uA(x) \in [0,1]$).
- Το σύνολο X περιλαμβάνει όλα τα αντικείμενα στα οποία μπορεί να γίνει αναφορά.
- uA(x): *βαθμός αληθείας* (*degree of truth*) - τιμές στο διάστημα [0,1].
- Η συνάρτηση uA ονομάζεται *συνάρτηση συγγένειας* (*membership function*).
- Προέλευση uA:
- Υποκειμενικές εκτιμήσεις
- Προκαθορισμένες (ad hoc) μορφές
- Συχνότητες εμφανίσεων και πιθανότητες
- Φυσικές μετρήσεις
- Διαδικασίες μάθησης και προσαρμογής (νευρωνικά δίκτυα)

Αναπαράσταση Ασαφών Συνόλων

- Αναλυτική έκφρασης της uA
- Απλούστευση: *τμηματικώς γραμμικής απεικόνιση* της uA
- Σύνολο ζευγών της μορφής uA(x)/x
- Π.χ. ψηλός = {0/1.7, 0/1.75, 0.33/1.8, 0.66/1.85, 1/1.9, 1/1.95}
- Με ζεύγη της μορφής (x, uA(x)):
- Π.χ. ψηλός = { (1.7, 0), (1.75, 0), (1.8, 0.33), (1.85, 0.66), (1.9, 1), (1.95, 1) }





Ασαφείς Σχέσεις (1/2)

- Ασαφή σύνολα ορισμένα σε πεδία αναφοράς ανώτερης διάστασης.
- **Παράδειγμα**: R = "x *είναι βαρύτερο από* y" $x \in X$, $y \in Y$ και $R \in X \times Y$
- Αναπαράσταση της R, σε μορφή πίνακα

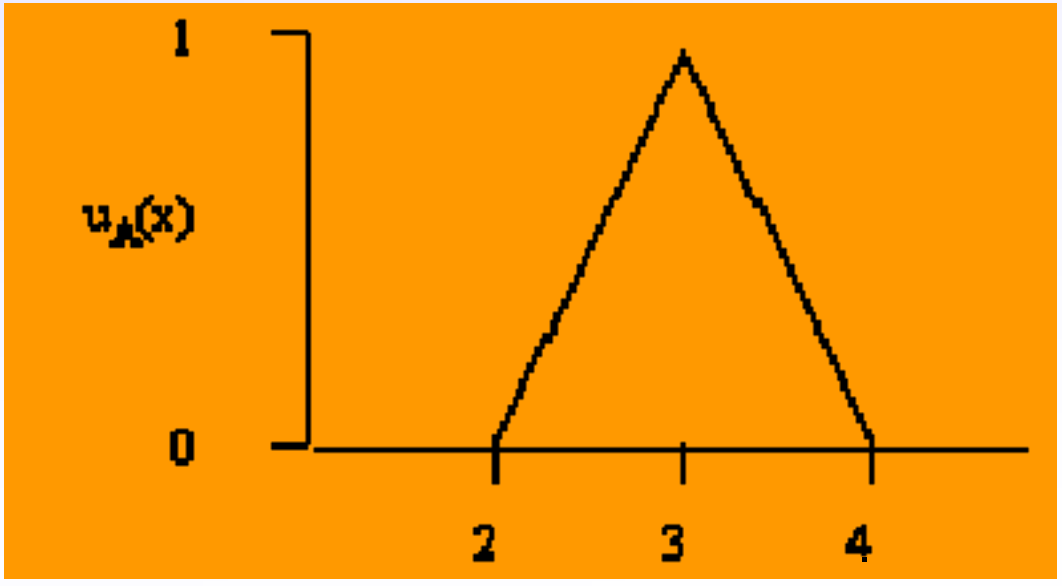
$$R = \begin{bmatrix} u_R(x_1, y_1) & u_R(x_1, y_2) & \cdots & u_R(x_1, y_n) \\ u_R(x_2, y_1) & u_R(x_2, y_2) & \cdots & u_R(x_2, y_n) \\ & \vdots & & \\ u_R(x_m, y_1) & u_R(x_m, y_2) & \cdots & u_R(x_m, y_n) \end{bmatrix}$$

Ασαφείς Σχέσεις (2/2)

- **Σύνθεση** (*composition*) Ασαφών Σχέσεων: συνδυασμός ασαφών σχέσεων.
- *Σύνθεση max-min* (*max-min composition*)
- *Σύνθεση max-product* (*max-product composition*).
- Αν R1(x,y) και R2(y,z) είναι δύο ασαφείς σχέσεις ορισμένες στα σύνολα X×Y και Y×Z αντίστοιχα, τότε η σύνθεσή τους δίνει μία νέα σχέση R1°R2 ορισμένη στο X×Z με συνάρτηση συγγένειας:
- *Σύνθεση max-min*: $u_{R1\circ R2}(x,z) = \bigvee_y [u_{R1}(x,y) \wedge u_{R2}(x,y)]$
- *Σύνθεση max-product*: $u_{R1\circ R2}(x,z) = \bigvee_y [u_{R1}(x,y) \bullet u_{R2}(x,y)]$

Ασαφείς Μεταβλητές και Ασαφείς Αριθμοί

- *Ασαφής Μεταβλητή* (*fuzzy variable*): οι τιμές τις ορίζονται με ασαφή σύνολα.
- Π.χ. τα ασαφή σύνολα {κοντός, μεσαίος, ψηλός} θα μπορούσαν να είναι το πεδίο τιμών της ασαφούς μεταβλητής "ύψος".
- "ύψος": *λεκτική* (*linguistic*) μεταβλητή.
- *Ασαφείς αριθμοί* (*fuzzy numbers*): ασαφή υποσύνολα του συνόλου των πραγματικών αριθμών. Π.χ. "Ασαφές 3" στο σχήμα.
- μη ασαφείς τιμές: *crisp* (*σαφείς, συγκεκριμένες*).



Η Αρχή της Επέκτασης (1/2)

- Επέκταση των εννοιών και των υπολογιστικών τεχνικών των κλασσικών μαθηματικών στο πλαίσιο των ασαφών.
- συνάρτηση *f*: ορίζει απεικόνιση του *X*={*x1, x2, .., xn*} στο *Y*={*y1, y2, .., yn*}, έτσι ώστε *y1=f*(*x1*), *y2=f*(*x2*), .., *yn=f*(*xn*).
- ασαφές σύνολο *A* ορισμένο στα στοιχεία του *X*
- A = { uA(x1)/x1 , uA(x2)/x2 , ... , uA(xn)/xn }
- Αν η είσοδος *x* της συνάρτησης *f* γίνει ασαφής μέσω του συνόλου *A*, τότε τι συμβαίνει με την έξοδο *y* ;
- **Αρχή Επέκτασης**: υπολογισμός ασαφούς συνόλου *B* με εφαρμογή της *f* στο *A*.
- B = f(A) = { uA(x1)/f(x1) , uA(x2)/f(x2) , ... , uA(xn)/f(xn) }
- δηλαδή, κάθε *yi=f*(*xi*) γίνεται ασαφές σε βαθμό *uA*(*xi*)
- πρακτικά, η *uB*(*y*) προκύπτει από την *uA*(*x*) όπου το *x* αντικαθίσταται με την έκφραση που προκύπτει για αυτό από την επίλυση της *f* ως προς *x*

Η Αρχή της Επέκτασης (2/2)

- Ειδικές Περιπτώσεις
- αν περισσότερα του ενός διαφορετικά *x* (έστω τα *xm* και *xn*) δίνουν μέσω της συνάρτησης *f* το ίδιο *y* (έστω το *y0*), τότε: uB(yo)=uA(xm) $\vee$ uA(xn).
- η μέγιστη τιμή συγγένειας των *xm* και *xn* στο *A* επιλέγεται ως βαθμός συγγένειας του *y0* στο *B*
- αν για κάποιο *yo* του *B* δεν υπάρχει *x0* του *A* τέτοιο ώστε *yo=f(xo)*, τότε η τιμή συγγένειας του *B* στο *yo* είναι μηδέν.
- Γενίκευση σε περισσότερες διαστάσεις:
- αν υπάρχουν οι μεταβλητές *u*, *v*, ..., *w* ορισμένες στα σύνολα *U*, *V*, ..., *W* αντίστοιχα
- *m* διαφορετικά ασαφή σύνολα *A1*, *A2*, .., *Am* ορισμένα στο $U \times V \times ... \times W$
- η πολυπαραμετρική συνάρτηση *y=f*(*u*,*x*,...,*w*)
- τότε η ασαφοποίηση του χώρου των *y*, δηλαδή η συνάρτηση συγγένειας του συνόλου *B*, ορίζεται ως εξής:

$$u_B(y) = \vee_{U \times V \times ... \times W} [u_{A1}(u) \wedge u_{A2}(v) \wedge ... \wedge u_{Am}(w)] / f(u, y, ..., w)$$

Παράδειγμα Χρήσης Αρχής Επέκτασης (1/2)

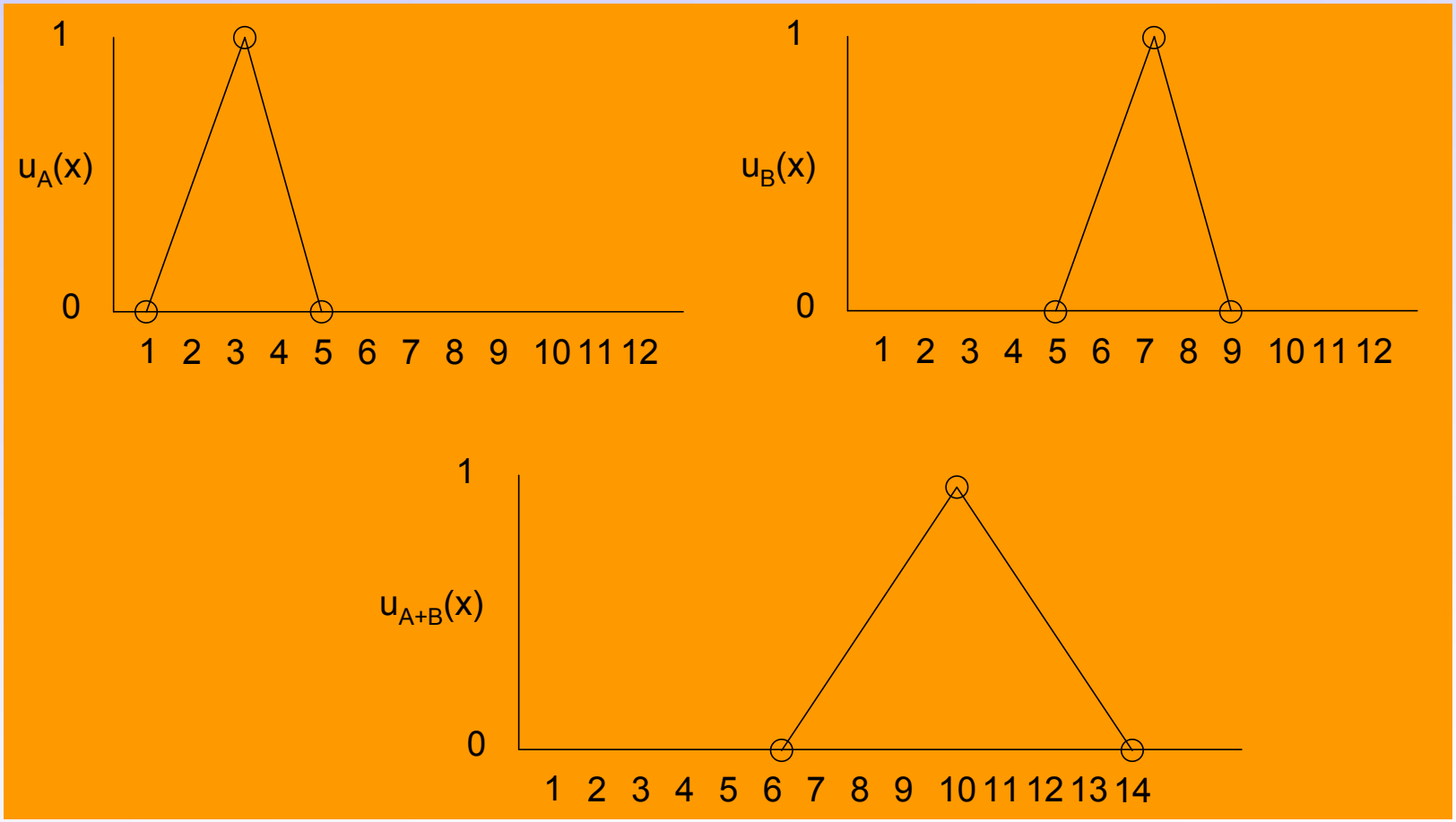
- Πρόσθεση των αριθμών A:"ασαφές 3" και B:"ασαφές 7"
- A: "ασαφές 3" = { 0/1, 0.5/2, 1/3, 0.5/4, 0/5 }
- B: "ασαφές 7" = { 0/5, 0.5/6, 1/7, 0.5/8, 0/9 }
- Κατασκευάζεται ο πίνακας:

B \ A	x=1	x=2	x=3	x=4	x=5
y=5	0 0	0.5 0	1 0	0.5 0	0 0
y=6	0 0.5	0.5 0.5	1 0.5	0.5 0.5	0 0.5
y=7	0 1	0.5 1	1 1	0.5 1	0 1
y=8	0 0.5	0.5 0.5	1 0.5	0.5 0.5	0 0.5
y=9	0 0	0.5 0	1 0	0.5 0	0 0

Παράδειγμα Χρήσης Αρχής Επέκτασης (2/2)

$$u_{C=A+B}(z) = \bigvee_{z=x+y} [u_A(x) \wedge u_B(y)]/(x+y)$$

- Έστω z=9. Υπάρχουν δύο συνδυασμοί *x* και *y* που μας δίνουν άθροισμα 9.
- αρχή της επέκτασης:
- uA+B(9) =max(min(uA(4), uB(5)), min(uA(3), uB(6)), min(uA(2), uB(7)) , min(uA(1), uB(8))) =
- max(min(0.5, 0), min(1, 0.5), min(0.5, 1), min(0, 0.5)) =
- max(0, 0.5, 0.5, 0) = 0.5
- Άρα: ο βαθμός συγγένειας του z=9 στο ασαφές σύνολο *C=A+B* είναι 0.5.
- όμοια για τα υπόλοιπα στοιχεία του πίνακα, οπότε προκύπτει:
- uA+B(z) = { 0/8, 0.5/9, 1/10, 0.5/11, 0/12 }


1
$u_A(x)$
0
1 2 3 4 5 6 7 8 9 10 11 12
1
$u_B(x)$
0
1 2 3 4 5 6 7 8 9 10 11 12
1
$u_{A+B}(x)$
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14

Ασαφείς Προτάσεις και Ασαφείς Κανόνες

- *Ασαφής πρόταση* είναι αυτή που θέτει μια τιμή σε μια ασαφή μεταβλητή.
- *Ασαφής κανόνας* (*fuzzy rule*): είναι μία υπό συνθήκη έκφραση που συσχετίζει δύο ή περισσότερες ασαφείς προτάσεις.
- "Εάν η ταχύτητα είναι μέτρια τότε η πίεση στα φρένα να είναι μέτρια"
- Η αναλυτική περιγραφή ενός ασαφούς κανόνα if-then είναι μία ασαφής σχέση R(x,y) που ονομάζεται *σχέση συνεπαγωγής* (*implication relation*).
- Γενική της μορφή της σχέσης (συνάρτησης) συνεπαγωγής:
- $R(x,y) \equiv u(x,y) = \varphi(uA(x), uB(y))$
- *φ*: *τελεστής συνεπαγωγής* (*implication operator*)

Μερικοί Ασαφείς Τελεστές Συνεπαγωγής

Ονομασία Τελεστή	**Αναλυτική Έκφραση του φ[uA(x),uB(y)]**
φ_m: Zadeh Max-Min	$(u_A(x) \wedge u_B(y)) \vee (1- u_A(x))$
φ_c: Mandani Min	$u_A(x) \wedge u_B(y)$
φ_p: Larsen Product	$u_A(x) \bullet u_B(y)$
φ_α: Arithmetic	$1 \wedge (1-u_A(x) + u_B(y))$
φ_b: Boolean	$(1-u_A(x)) \vee u_B(y))$

Συλλογιστικές Διαδικασίας GMP *και* GMT

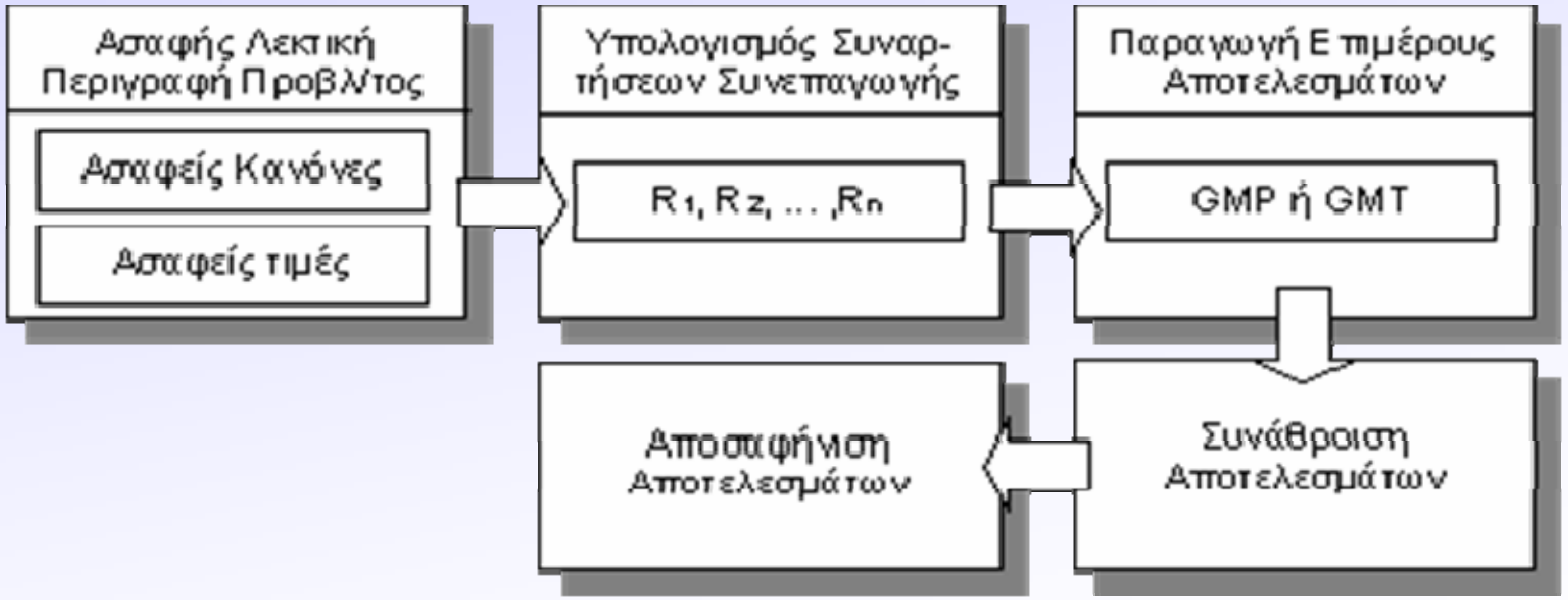
- Γενική μορφή προβλημάτων κατά τη συλλογιστική με ασαφείς κανόνες:
- if x is A then y is B
- x is A' y is B' (?)
- μέσω της συλλογιστικής διαδικασίας GMP (Generalized Modus Ponens - GMP): B'=A'oR(*x,y*)
- if x is A then y is B
- x is A' (?) y is B'
- μέσω της συλλογιστικής διαδικασίας GMT (Generalized Modus Tollens - GMT): A'=R(*x,y*)oB'
- Η *σχέση συνεπαγωγής* R(x,y) που έχει επιλεγεί να χρησιμοποιηθεί, πρέπει να συνδυαστεί (σύνθεση) με την κατά περίπτωση γνωστή παράμετρο (A' ή B') ώστε να υπολογιστεί η άγνωστη παράμετρος.

Σύνοψη Ασαφούς Συλλογιστικής Διαδικασίας

- Με βάση έναν ασαφή κανόνα της μορφής:
- "*if x is A then y is B*"
- και έστω συλλογιστική διαδικασία GMP (δηλαδή γνωστό το *A'* ως τιμή του *x* και ζητούμενο το *B'* ως τιμή του *y*), τα ασαφή σύνολα *A* και *B* συνδυάζονται με κάποιον από τους *τελεστές συνεπαγωγής* και παράγουν τη *σχέση συνεπαγωγής* *R*(*x*,*y*).
- Από την *R*(*x*,*y*) μέσω σύνθεσης (έστω *max-min σύνθεση*) με το A' προκύπτει η άγνωστη ποσότητα *B'*:
- B'=A'oR(*x*,*y*)

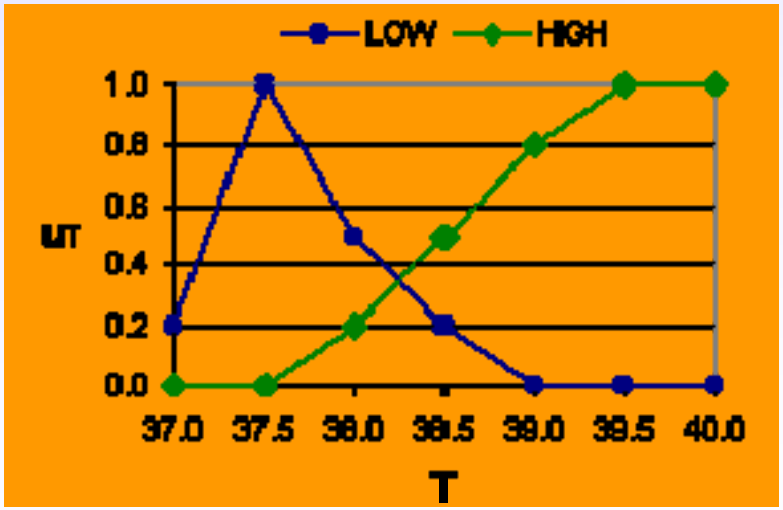
Ασαφής Συλλογιστική

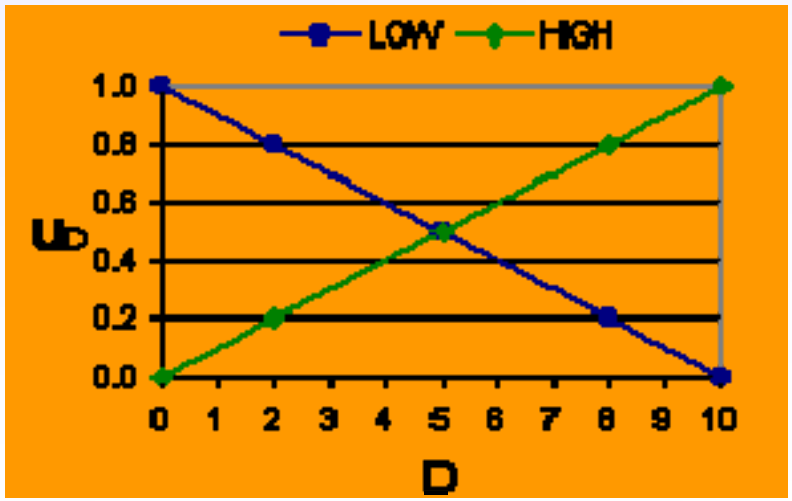
- Εξαγωγή συμπερασμάτων με χρήση ασαφών κανόνων.
- Τέσσερα στάδια:
- Υπολογισμός της συνάρτησης συνεπαγωγής για κάθε εμπλεκόμενο κανόνα.
- Παραγωγή επιμέρους αποτελεσμάτων μέσω κάποιας συλλογιστικής διαδικασίας.
- Συνάθροιση των επιμέρους αποτελεσμάτων.
- Αποσαφήνιση αποτελεσμάτων.



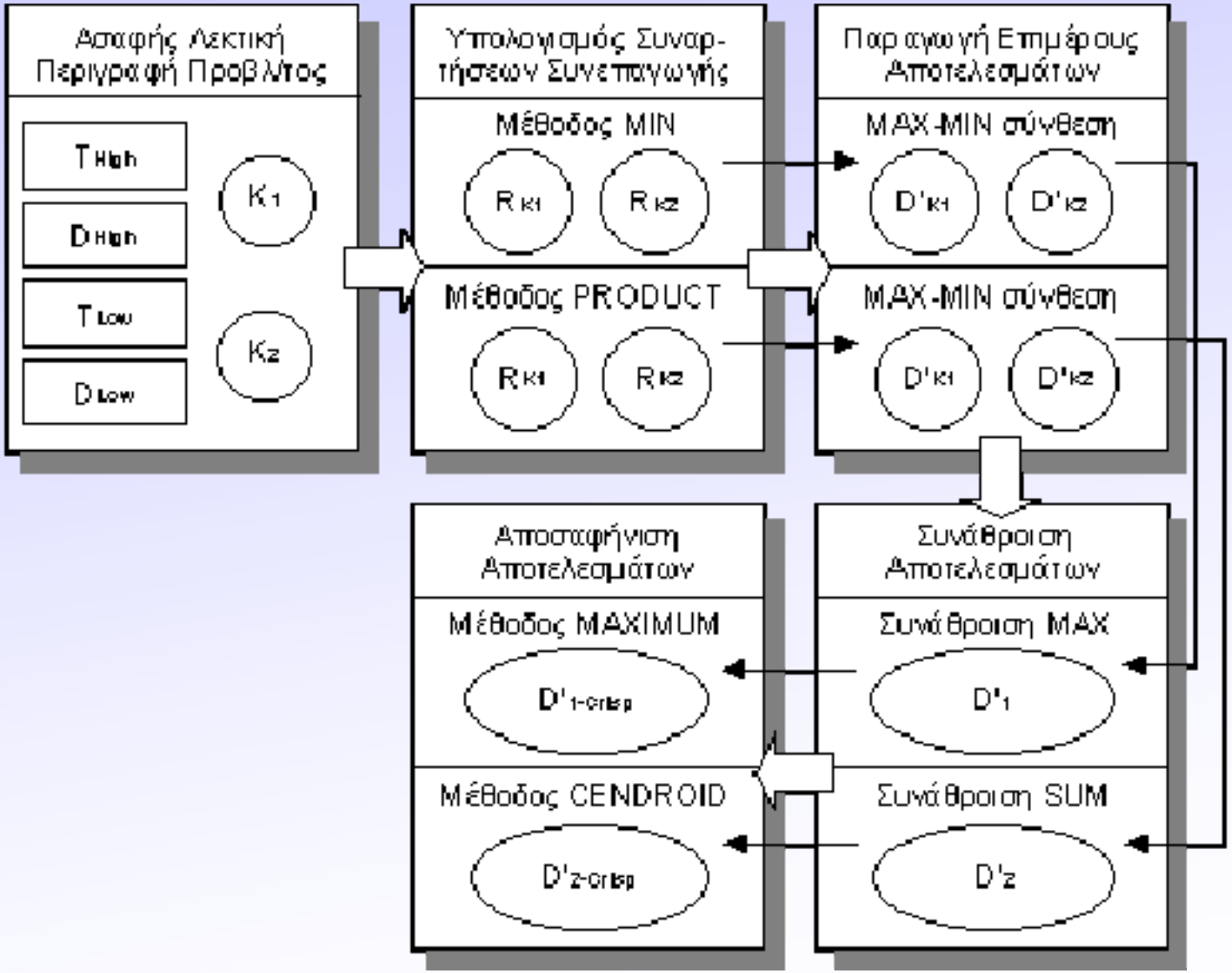
Παράδειγμα Προβλήματος Ασαφούς Συλλογιστικής (1/2)

- Έστω σύστημα που ρυθμίζει τη δόση D μιας φαρμακευτικής ουσίας που πρέπει να χορηγηθεί σε ασθενή, με βάση τη θερμοκρασία του T.
- Έστω ότι το σύστημα βασίζεται στους εξής δύο ασαφείς κανόνες:
- K1: if T is HIGH then D is HIGH
- K2: if T is LOW then D is LOW
- Δίνονται επίσης τα ασαφή σύνολα HIGH και LOW για τα μεγέθη T και D:
- TLOW = { 0.2/37, 1/37.5, 0.5/38, 0.2/38.5, 0/39, 0/39.5, 0/40 }
- THIGH = { 0/37, 0/37.5, 0.2/38, 0.5/38.5, 0.8/39, 1/39.5, 1/40 }
- DLOW = { 1/0, 0.8/2, 0.5/5, 0.2/8, 0/10 }
- DHIGH = { 0/0, 0.2/2, 0.5/5, 0.8/8, 1/10 }
- Αν T'=38.5, να υπολογιστεί η τιμή του D' με συλλογιστική διαδικασία GMP.





Παράδειγμα Προβλήματος Ασαφούς Συλλογιστικής (2/2)



Βήμα Α1: Υπολογισμός συνάρτησης συνεπαγωγής

- Μέθοδος MIN
- 2 κανόνες: δύο τελεστές συνεπαγωγής, οι RK1 και RK2.
- Χρησιμοποιείται ο τελεστής συνεπαγωγής Mandani min (ή απλά MIN).
- **K1**: if T is HIGH then D is HIGH **K2:** if T is LOW then D is LOW
- Έστω ο K1. Κατασκευάζεται ο πίνακας αριστερά:
- Κάθε κελί του εσωτερικού πίνακα περιέχει το min(uTHIGH, uDHIGH) για τα T και D της γραμμής και στήλης στην οποία βρίσκεται.
- Όμοια προκύπτει και η RK2(TLOW,DLOW) για τον κανόνα K2 (πίνακας δεξιά)
- **Γενίκευση**: αν N εκφράσεις στο if τμήμα τότε προκύπτει πίνακας N+1 διαστάσεων.

R_{K1}	**D**	0	2	5	8	10
T		0	0.2	0.5	0.8	1
37.0	0	0	0	0	0	0
37.5	0	0	0	0	0	0
38.0	0.2	0	0.2	0.2	0.2	0.2
38.5	0.5	0	0.2	0.5	0.5	0.5
39.0	0.8	0	0.2	0.5	0.8	0.8
39.5	1	0	0.2	0.5	0.8	1
40.0	1	0	0.2	0.5	0.8	1

R_{K2}	**D**	0	2	5	8	10
T		1	0.8	0.5	0.2	0
37.0	0.2	0.2	0.2	0.2	0.2	0
37.5	1	1	0.8	0.5	0.2	0
38.0	0.5	0.5	0.5	0.5	0.2	0
38.5	0.2	0.2	0.2	0.2	0.2	0
39.0	0	0	0	0	0	0
39.5	0	0	0	0	0	0
40.0	0	0	0	0	0	0

Βήμα Α2: Παραγωγή επιμέρους αποτελεσμάτων (1/2)

- Με εφαρμογή της συλλογιστικής διαδικασίας GMP:
- Κανόνας Κ1: D΄Κ1=T'oRK1(THIGH,DHIGH)
- Κανόνας Κ2: D΄Κ2=T'oRK2(TLOW,DLOW)
- Απαιτείται η γραφή της θερμοκρασίας T'=38.5 σε μορφή ασαφούς συνόλου, δηλαδή:
- T' = 38.5 = { 0/37, 0/37.5, 0/38, 1/38.5, 0/39, 0/39.5, 0/40 }
- Χρησιμοποιείται η μέθοδος *σύνθεσης* (o) max-min (η συνηθέστερη περίπτωση).
- Τεχνική όμοια με πολλαπλασιασμό πινάκων: χρησιμοποιείται min αντί πολλαπλασιασμού και max αντί πρόσθεσης.
- 1ος πίνακας το ασαφές σύνολο T' (1x7) και 2ος ο αριστερά του βήματος Α1 (7x5)
- Το αποτέλεσμα θα είναι ένας πίνακας 1x5 που θα αποτελεί και την ποσότητα D'K1.

$$D'_{K1} = [0/37,\ 0/37.5,\ 0/38,\ 1/38.5,\ 0/39,\ 0/39.5,\ 0/40] \quad o$$

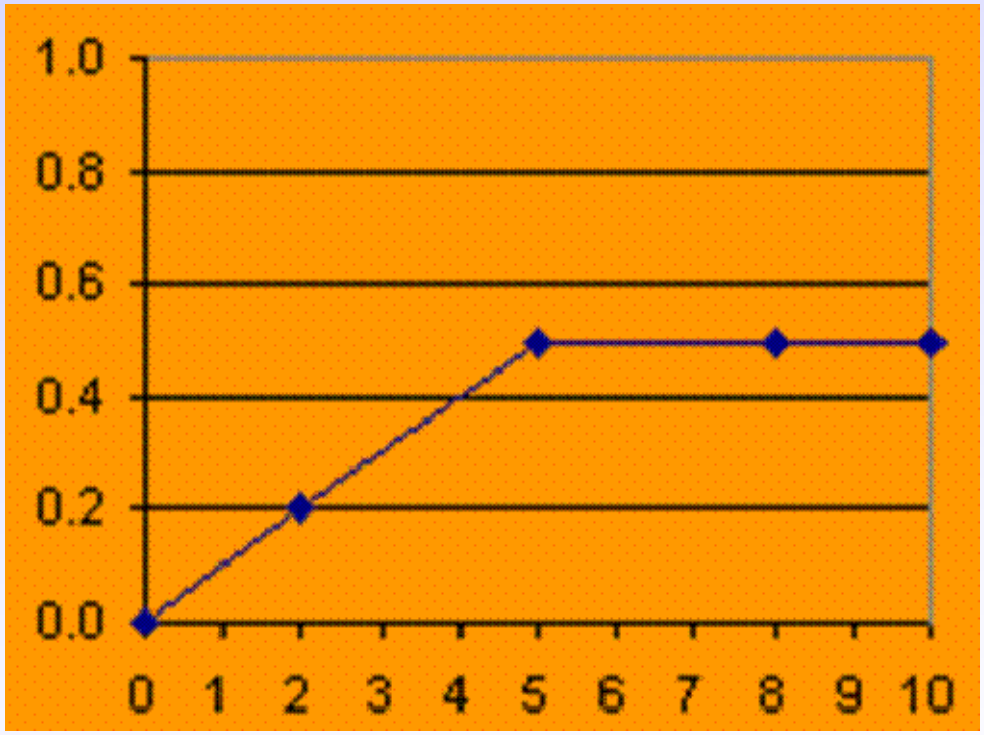
T \ D	0	2	5	8	10
37	0	0	0	0	0
37.5	0	0	0	0	0
38	0	0.2	0.2	0.2	0.2
38.5	0	0.2	0.5	0.5	0.5
39	0	0.2	0.5	0.8	0.8
39.5	0	0.2	0.5	0.8	1
40	0	0.2	0.5	0.8	1

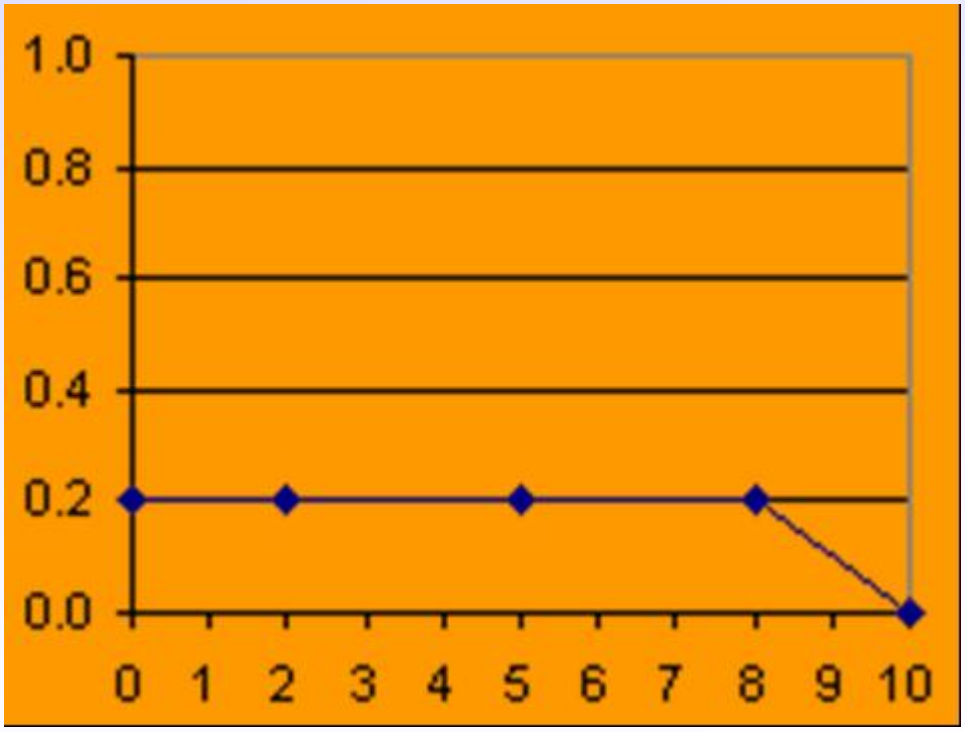
$\Rightarrow$

D'K1 = { 0/0, 0.2/2, 0.5/5, 0.5/8, 0.5/10 }

Βήμα Α2: Παραγωγή επιμέρους αποτελεσμάτων (2/2)

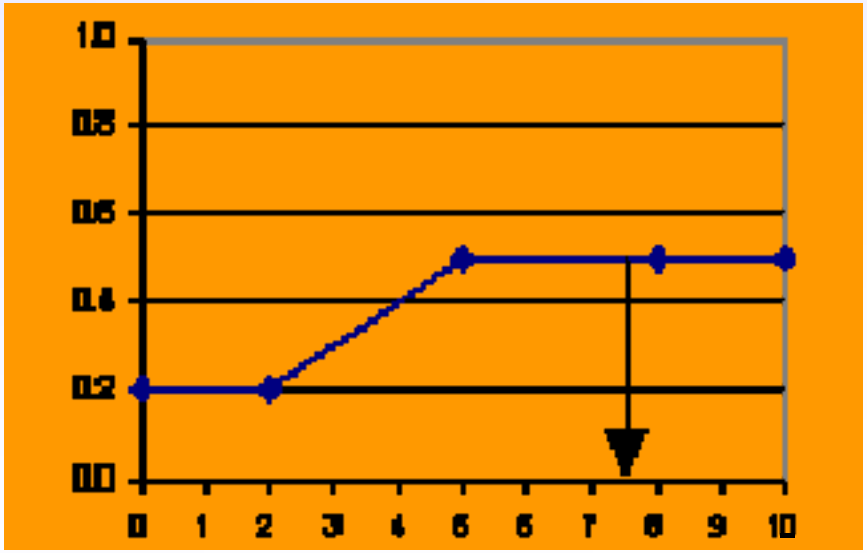
- Όμοια, ο κανόνας Κ2 δίνει: D'K2 = { 0.2/0, 0.2/2, 0.2/5, 0.2/8, 0/10 }
- D'K1 (αριστερά) και D'K2 (δεξιά).





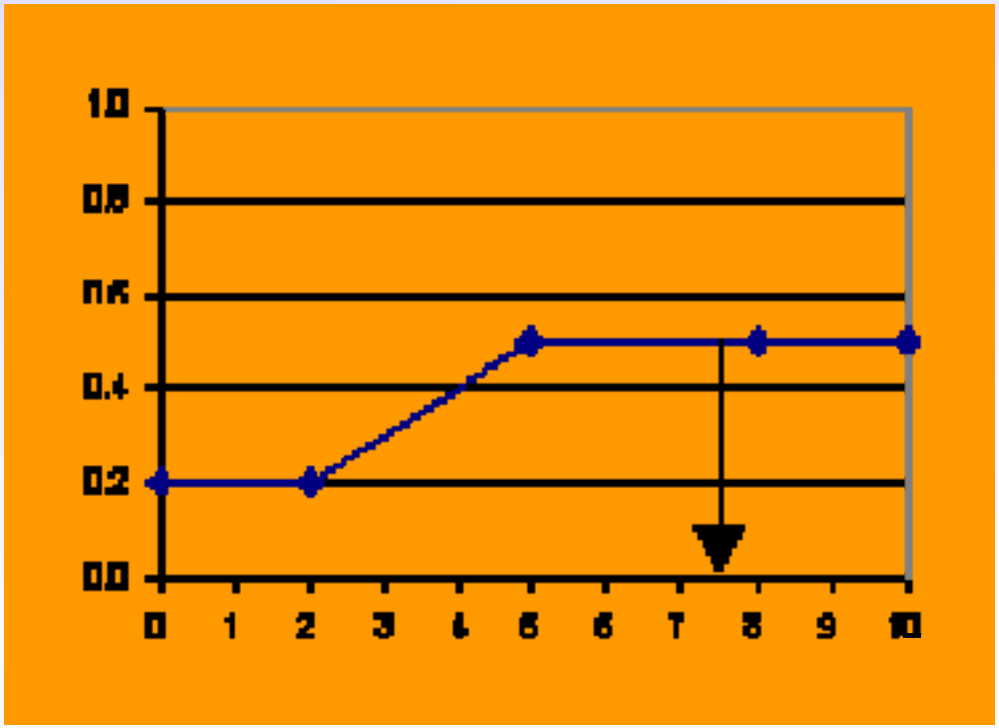
Βήμα Α3: Συνάθροιση αποτελεσμάτων

- Μέθοδος MAX
- Υπολογίζει τη συνδυασμένη έξοδο των κανόνων παίρνοντας τη μέγιστη τιμή συγγένειας από τις παραμέτρους εξόδου κάθε κανόνα, σημείο προς σημείο (*pointwise maximum* - maxp/w).
- Δεδομένου ότι έχει υπολογιστεί:
- D1'K1 = { 0/0, 0.2/2, 0.5/5, 0.5/8, 0.5/10 }
- D1'K2 = { 0.2/0, 0.2/2, 0.2/5, 0.2/8, 0/10 }
- η συνάθροισή τους κατά *MAX* δίνει
- D1'= {max(0,0.2)/0, max(0.2,0.2)/2, max(0.5,0.2)/5, max(0.5,0.2)/8, max(0.5,0)/10} =
- = { 0.2/0, 0.2/2, 0.5/5, 0.5/8, 0.5/10 }



Βήμα Α4: Αποσαφήνιση

- Μέθοδος αποσαφήνισης MAXIMUM
- διακριτή τιμή: μέγιστη τιμή συγγένειας του τελικού αποτελέσματος.
- με average-of-maxima αποσαφήνιση: D1= (5+8+10)/3=7.7



Βήμα Β1: Υπολογισμός συνάρτησης συνεπαγωγής

- Μέθοδος PRODUCT
- δύο κανόνες: δύο τελεστές συνεπαγωγής, οι RK1 και RK2. Έστω ο τελεστής συνεπαγωγής Larsen Product (ή απλά PRODUCT).
- **K1**: if T is HIGH then D is HIGH **K2**: if T is LOW then D is LOW
- **Κανόνας K1**. Κατασκευάζεται ο πίνακας αριστερά: Κάθε κελί του εσωτερικού πίνακα (σχέση συνεπαγωγής RK1) περιέχει το (uTHIGH • uDHIGH) για τα T και D της γραμμής και στήλης στην οποία βρίσκεται.
- Όμοια προκύπτει και η RK2(TLOW,DLOW) για τον κανόνα K2 (πίνακας δεξιά)

R_{K1}	**D**	0	2	5	8	10
T		0	0.2	0.5	0.8	1
37	0	0	0	0	0	0
37.5	0	0	0	0	0	0
38	0.2	0	0.04	0.1	0.16	0.2
38.5	0.5	0	0.1	0.25	0.4	0.5
39	0.8	0	0.16	0.4	0.64	0.8
39.5	1	0	0.2	0.5	0.8	1
40	1	0	0.2	0.5	0.8	1

R_{K2}	**D**	0	2	5	8	10
T		1	0.8	0.5	0.2	0
37	0.2	0.2	0.16	0.1	0.04	0
37.5	1	1	0.8	0.5	0.2	0
38	0.5	0.5	0.4	0.25	0.1	0
38.5	0.2	0.2	0.16	0.1	0.04	0
39	0	0	0	0	0	0
39.5	0	0	0	0	0	0
40	0	0	0	0	0	0

Βήμα Β2: Παραγωγή επιμέρους αποτελεσμάτων (1/2)

- Με εφαρμογή της συλλογιστικής διαδικασίας GMP:
- Κανόνας Κ1: D΄Κ1=T'oRK1(THIGH,DHIGH)
- Κανόνας Κ2: D΄Κ2=T'oRK2(TLOW,DLOW)
- Απαιτείται η γραφή της θερμοκρασίας T'=38.5 σε μορφή ασαφούς συνόλου:
- T' = 38.5 = { 0/37, 0/37.5, 0/38, 1/38.5, 0/39, 0/39.5, 0/40 }
- Χρησιμοποιείται η μέθοδος *σύνθεσης* (o) max-min (η συνηθέστερη περίπτωση).
- 1ος πίνακας το ασαφές σύνολο T' (1x7) και 2ος ο εσωτερικός του βήματος Β1 (7x5)
- Το αποτέλεσμα θα είναι ένας πίνακας 1x5 που θα αποτελεί και την ποσότητα D'K1.

D'_{K1}= [0/37, 0/37.5, 0/38, 1/38.5, 0/39, 0/39.5, 0/40] o

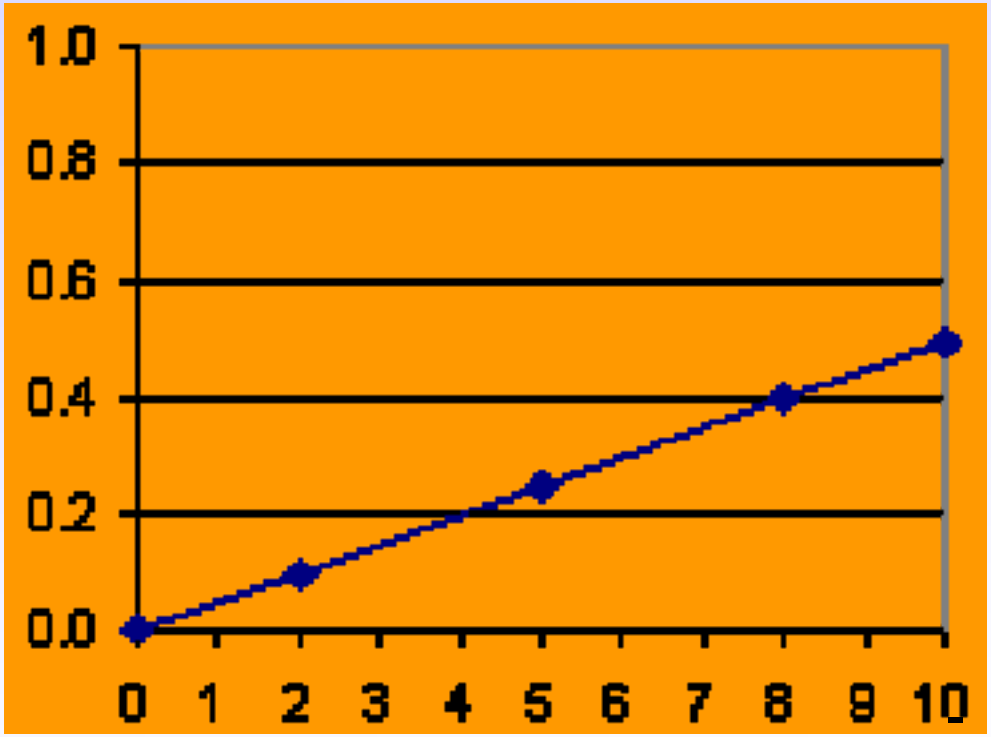
T \ D	0	2	5	8	10
37	0	0	0	0	0
37.5	0	0	0	0	0
38	0	0.04	0.1	0.16	0.2
38.5	0	0.1	0.25	0.4	0.5
39	0	0.16	0.4	0.64	0.8
39.5	0	0.2	0.5	0.8	1
40	0	0.2	0.5	0.8	1

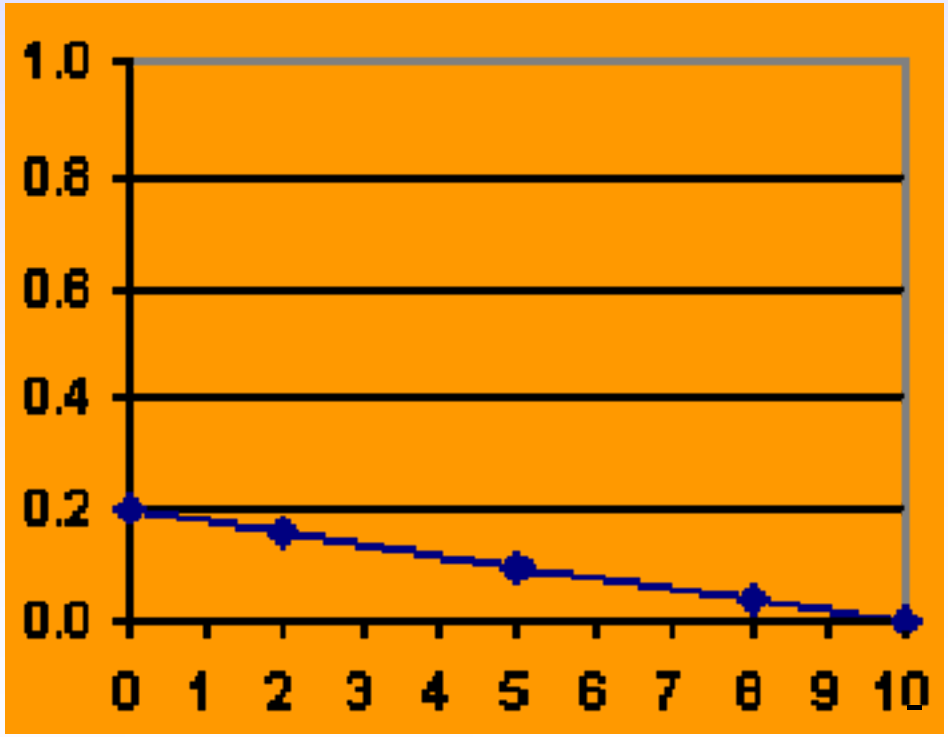
⇒

D'K1 = { 0/0, 0.1/2, 0.25/5, 0.4/8, 0.5/10 }

Βήμα Β2: Παραγωγή επιμέρους αποτελεσμάτων (2/2)

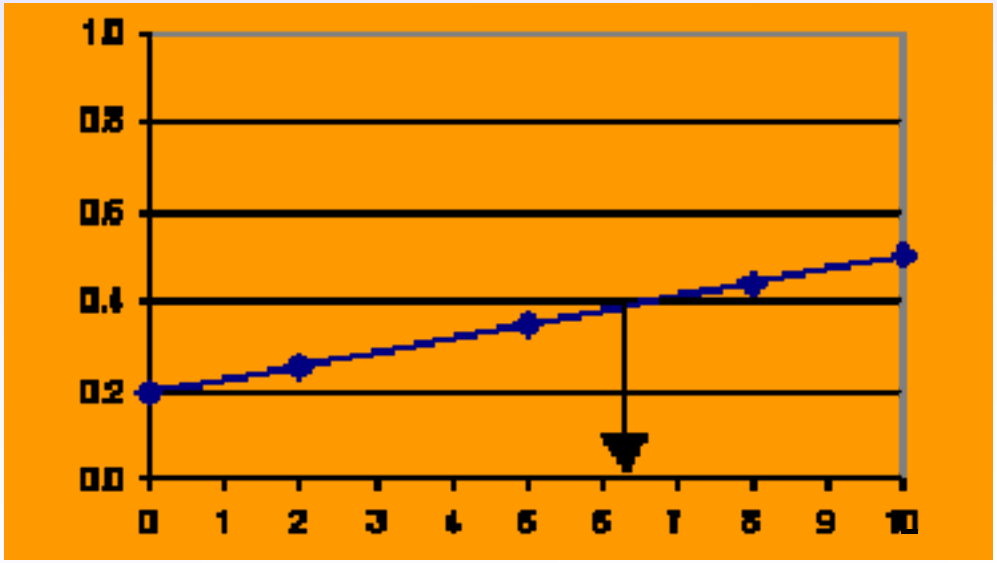
- Όμοια προκύπτει ότι ο κανόνας Κ2 δίνει: D'K2 = { 0.2/0, 0.16/2, 0.1/5, 0.04/8, 0/10 }
- Γραφική απεικόνιση των D'K1 (αριστερά) και D'K2 (δεξιά).





Βήμα Β3: Συνάθροιση αποτελεσμάτων

- Μέθοδος SUM
- Υπολογίζει τη συνδυασμένη έξοδο των κανόνων παίρνοντας το άθροισμα των τιμών συγγένειας των παραμέτρων εξόδου κάθε κανόνα, σημείο προς σημείο (*pointwise sum* - sump/w).
- Δεδομένου ότι έχει υπολογιστεί:
- D2'K1 = { 0/0, 0.1/2, 0.25/5, 0.4/8, 0.5/10 }
- D2'K2 = { 0.2/0, 0.16/2, 0.1/5, 0.04/8, 0/10 }
- ...η συνάθροισή τους κατά *MAX* δίνει....
- D2'= { (0+0.2)/0, (0.1+0.16)/2, (0.25+0.1)/5, (0.4+0.04)/8, (0.5+0)/10} = { 0.2/0, 0.26/2, 0.35/5, 0.44/8, 0.5/10 }



Βήμα Β4: Αποσαφήνιση

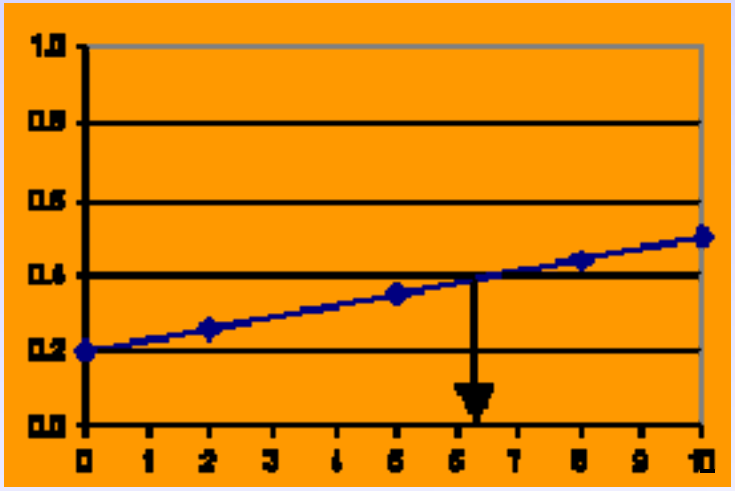
- Μέθοδος αποσαφήνισης CENDROID
- Η διακριτή τιμή είναι αυτή που προκύπτει από το κέντρο βάρους της τελικής συνάρτησης συγγένειας για την ασαφή παράμετρο εξόδου.
- Το κέντρο βάρους επιφάνειας που ορίζεται από μία συνάρτηση *f*(*t*): σχέση (1)

$$t_{\kappa\beta} = \frac{\int_{t1}^{t2} t \cdot f(t)dt}{\int_{t2}^{t1} f(t)dt}$$

- Για διακριτού συνόλου αναφοράς: διακριτό άθροισμα με δειγματοληψία *N* σημείων (σχ.2).

$$t_{\kappa\beta} = \frac{\sum_{i=1}^{N} t_i \cdot u_{OUT}(t_i)}{\sum_{i=1}^{N} u_{OUT}(t_i)}$$

- Με CENDROID αποσαφήνιση στα αποτελέσματα της συνάθροισης SUM, προκύπτει:



$$t_{D2'} = \frac{\sum_{i=1}^{5} t_i \cdot u_{D2'}(t_i)}{\sum_{i=1}^{5} u_{D2'}(t_i)} = \frac{0 \cdot 0.2 + 2 \cdot 0.26 + 5 \cdot 0.35 + 8 \cdot 0.44 + 10 \cdot 0.5}{0.2 + 0.26 + 0.35 + 0.44 + 0.5} = 6.2$$

Λήψη Αποφάσεων στο Σημασιολογικό Ιστό

Towards a Semantic Web

- The current Web represents information using
  - natural language (English, Hungarian, Chinese,…)
  - graphics, multimedia, page layout
- Humans can process this easily
  - can deduce facts from partial information
  - can create mental associations
  - are used to various sensory information
    - (well, sort of… people with disabilities may have serious problems on the Web with rich media!)

Towards a Semantic Web

- Tasks often require to combine data on the Web:
  - hotel and travel information may come from different sites
  - searches in different digital libraries
  - etc.
- Again, humans combine these information easily
  - even if different terminologies are used!

However…

- However: machines are ignorant!
  - partial information is unusable
  - difficult to make sense from, e.g., an image
  - drawing analogies automatically is difficult
  - difficult to combine information automatically
    - is **<foo:creator>** same as **<bar:author>**?
  - …

Example: automatic airline reservation

- Your automatic airline reservation
  - knows about your preferences
  - builds up knowledge base using your past
  - can combine the local knowledge with remote services:
    - airline preferences
    - dietary requirements
    - calendaring
    - etc
- It communicates with remote information

Example: data(base) integration

- Databases are very different in structure, in content
- Lots of applications require managing several databases
  - after company mergers
  - combination of administrative data for e-Government
  - biochemical, genetic, pharmaceutical research
  - etc.
- Most of these data are accessible from the Web (though not necessarily public yet)

And the problem is real...

CoCoDat - Collation of Cortical Data - Mozilla Firefox

File Edit View History Bookmarks Tools Help

CoCoMac DATABASES ORT EXAMPLES

CoCoDat: Collation of Cortical [s
microcircuitry] Data

CoCoDat is a microcircuitry database that
published experimental reports. The data
and cellular compartment), as well as the

- Morphology
- Firing properties
- Ionic currents
- Ionic conductances
- Synaptic currents
- Connectivity

The database is available for download u
data tables but also a Search Board with
manual or automatic relaxation of the sea

- Brain region
- Layer
- Neuron type

http://www.cocomac.org/cocodat/catalyzer/inc

Cell Centered Database - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://ccdb.ucsd.edu/sand/main?event=gallery&action=show&dpl=y

red database

Cell Centered Database™
National Center for Microscopy and Imaging Research
Gallery

Data | Search | Gallery | Dictionary | Publications | MyCCDB | Data Download | Contact us | Help

2D image | Reconstruction | Segmentation | Animation

y 23
e 071202tjd
: P1188
t Brain Maps
t Tom Deerinck
t BioRad RTS 2000MP
: Multiphoton
: montage of cerebellar cortex triple labeled for Hoescht stain (blue), IP3 receptor (green) and GFAP (red)

logged out

NeuronDB = Thalamic relay neuron - Overview (A) () - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://senselab.med.yale.edu

senselab

NeuronDB

Back

Thalamic relay neuron

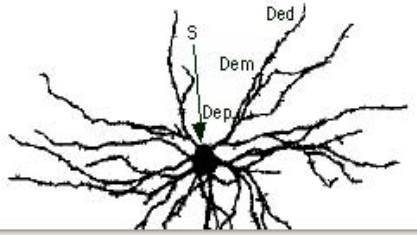
Mode: Overview | Data/Search | plus Connectivity | plus Classical References/Notes | Models

Region: Distal equivalent dendrite | Middle equivalent dendrite | Proximal equivalent dendrite | Soma | Axon hillock | Axon fiber | Axon terminal | All Compartments

Properties: Receptors | Channels | Transmitters | All Properties

Interoperation: Gene and Chromosome | Experimental Data (neurodatabase.org) | Microscopy Data (CCDB)

Neuron type: principal
Organism: Vertebrates



1. Equivalent dendrite — Show other
2. Distal equivalent dendrite — Show other
3. Middle equivalent dendrite — Show other
4. Proximal equivalent dendrite — Show other
5. Soma — Show other

Done

logged out

Example: digital libraries

- It means catalogues on the Web
  - librarians have known how to do that for centuries
  - goal is to have this on the Web, World-wide
  - extend it to multimedia data, too
- But it is more: software agents should also be librarians!
  - help you in finding the right publications

Example: semantics of Web Services

- Web services technology is great
- But if services are ubiquitous, searching issue comes up, for example:
  - “find me the best differential equation solver”
  - “check if it can be combined with the XYZ plotter service”
- It is necessary to characterize the service
  - not only in terms of input and output parameters…
  - …but also in terms of its semantics

What is needed?

- (Some) data should be available for machines for further processing
- Data should be possibly combined, merged on a Web scale
- Sometimes, data may describe other data…
- … but sometimes the data is to be exchanged by itself, like my calendar or my travel preferences
- Machines may also need to *reason* about that data

In short: we need a Web of Data!

In what follows…

- We will use a simplistic example to introduce the main Semantic Web concepts

- We take, as an **example** area, **data integration**

The rough structure of data integration

1. Map the various data onto an abstract data representation
   - make the data independent of its internal representation…
2. Merge the resulting representations
3. Start making queries on the whole!
   - queries not possible on the individual data sets

A *simplified* book store data (dataset "A")

Dataset A : RDBMS

Table : Books

ID	Author	Title	Publisher	Year
ISBN0-00-651409-X	id_xyz	The Glass Palace	id_qpr	2000

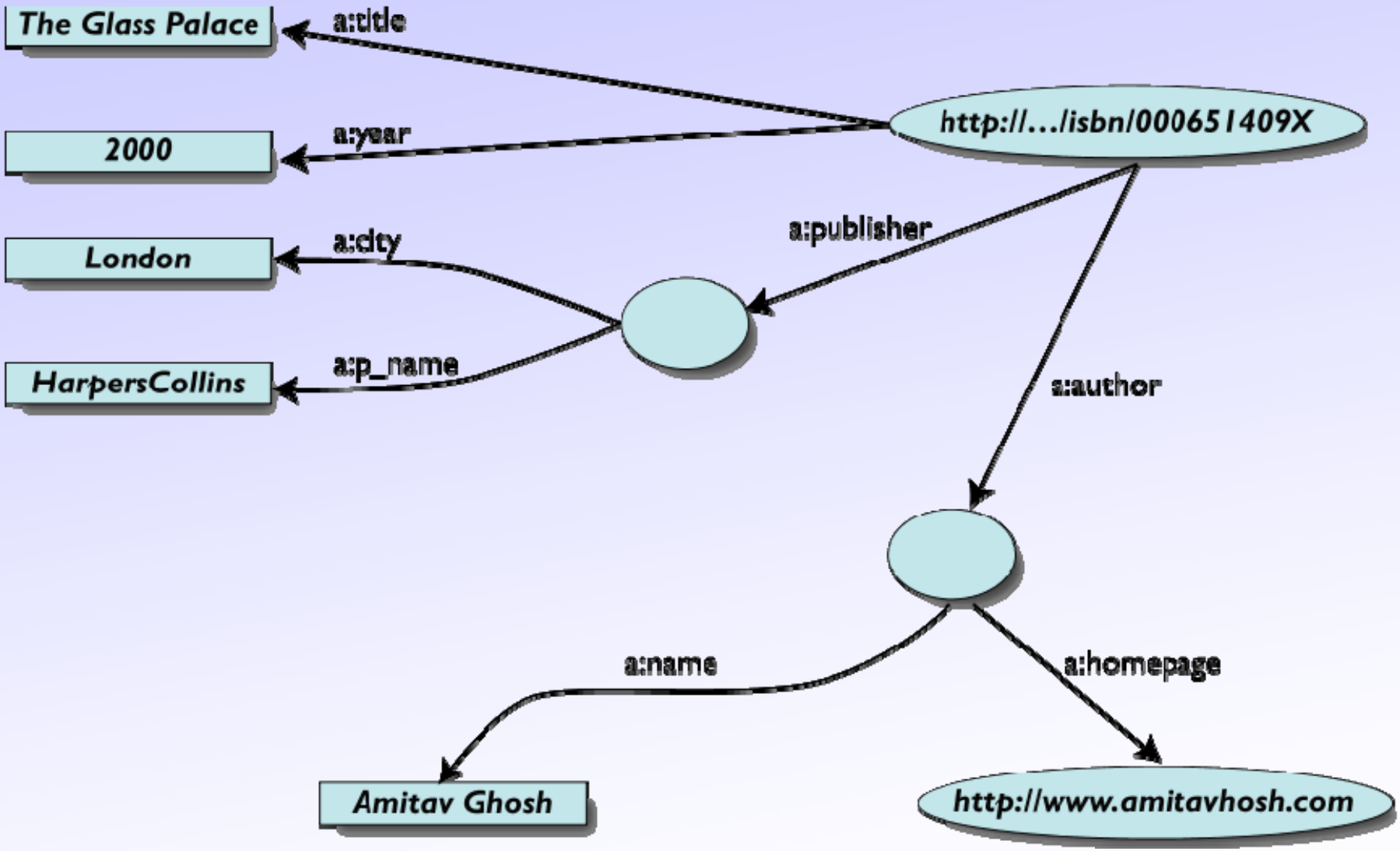
Table : Authors

ID	Name	Home Page
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

Table : Publishers

ID	Publ. Name	City
id_qpr	Harpers Collins	London

1st: export your data as a set of relations



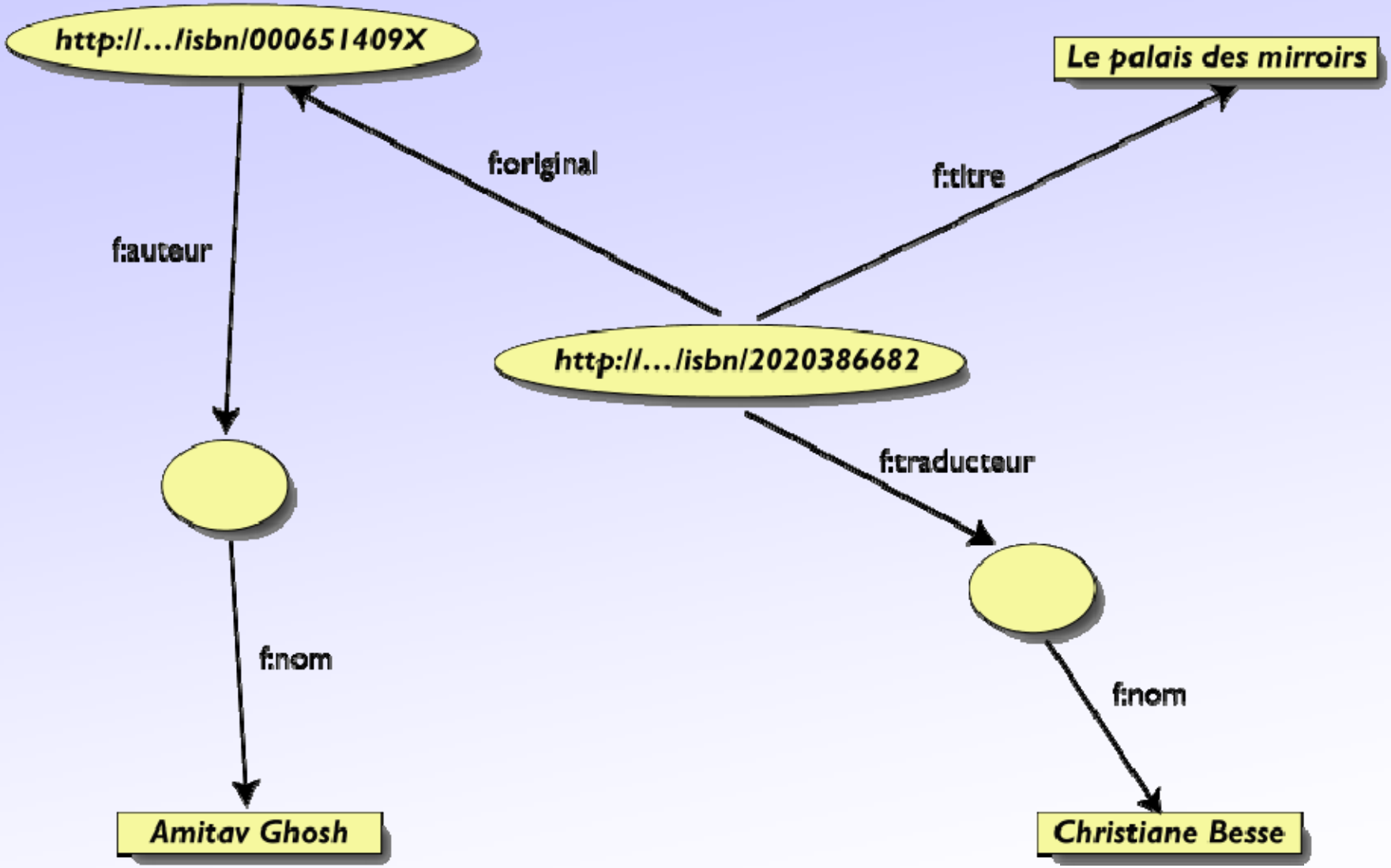
Another *book store data (dataset "F")*

Dataset F : Excel file

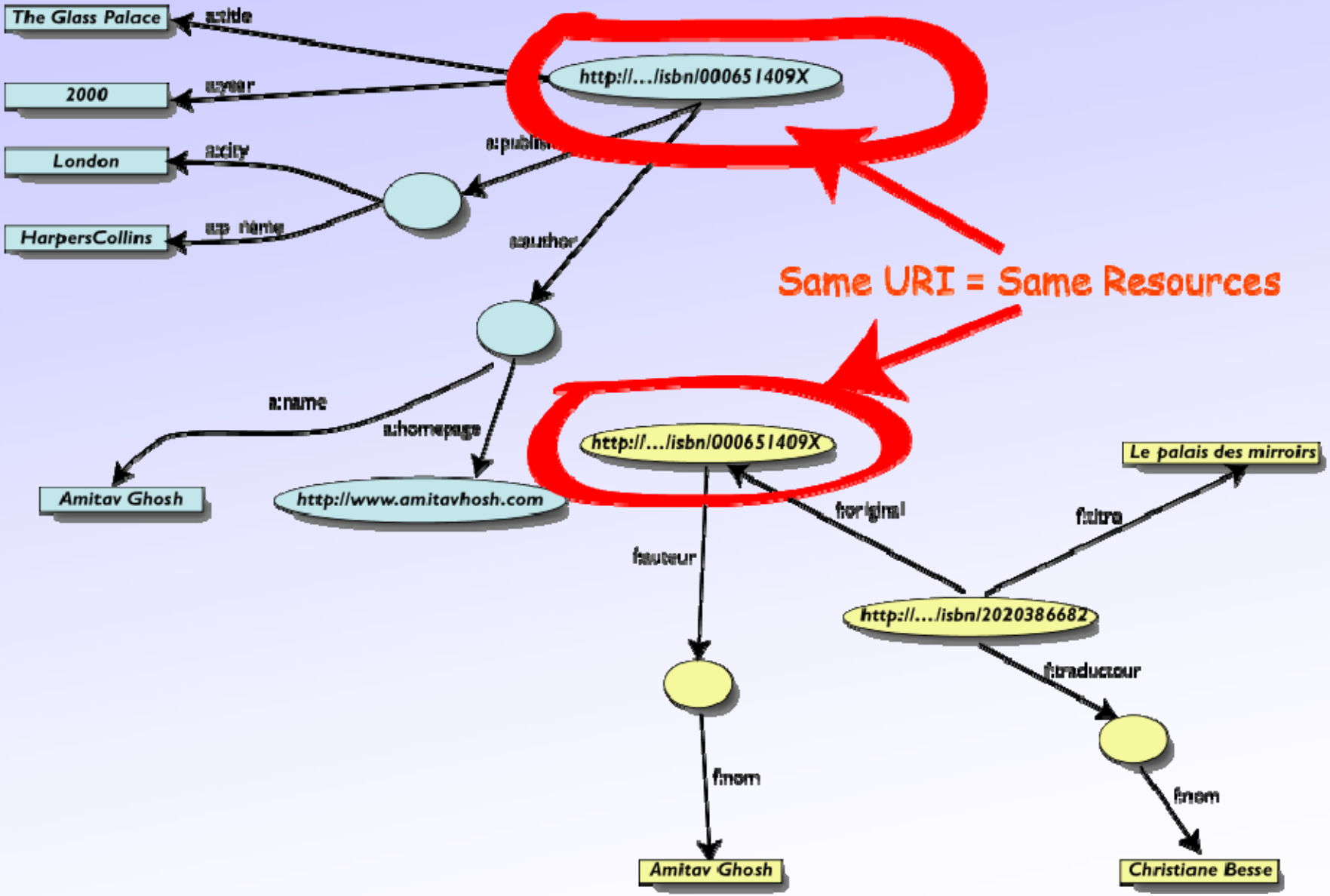
	A	B	C	D	E
1	**ID**	**Titre**	**Auteur**	**Traducteur**	**Original**
2	ISBN0 2020386682	Le Palais des miroirs	A7	A8	ISBN-0-00-651409-X
3					
4					
5					

	Nom
6	**Nom**
7	Ghosh, Amitav
8	Besse, Christianne

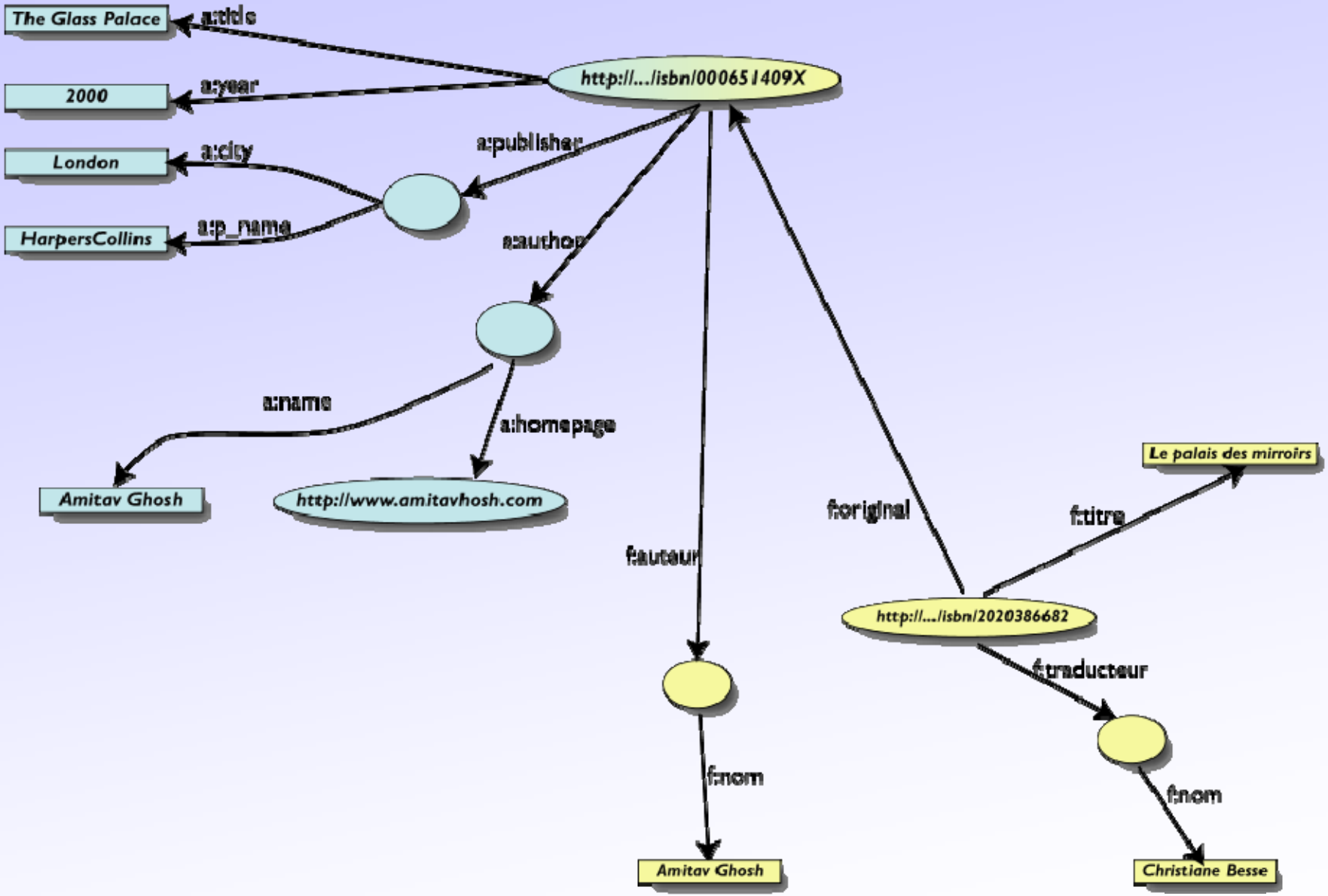
2nd: export your second set of data



3rd: start merging your data

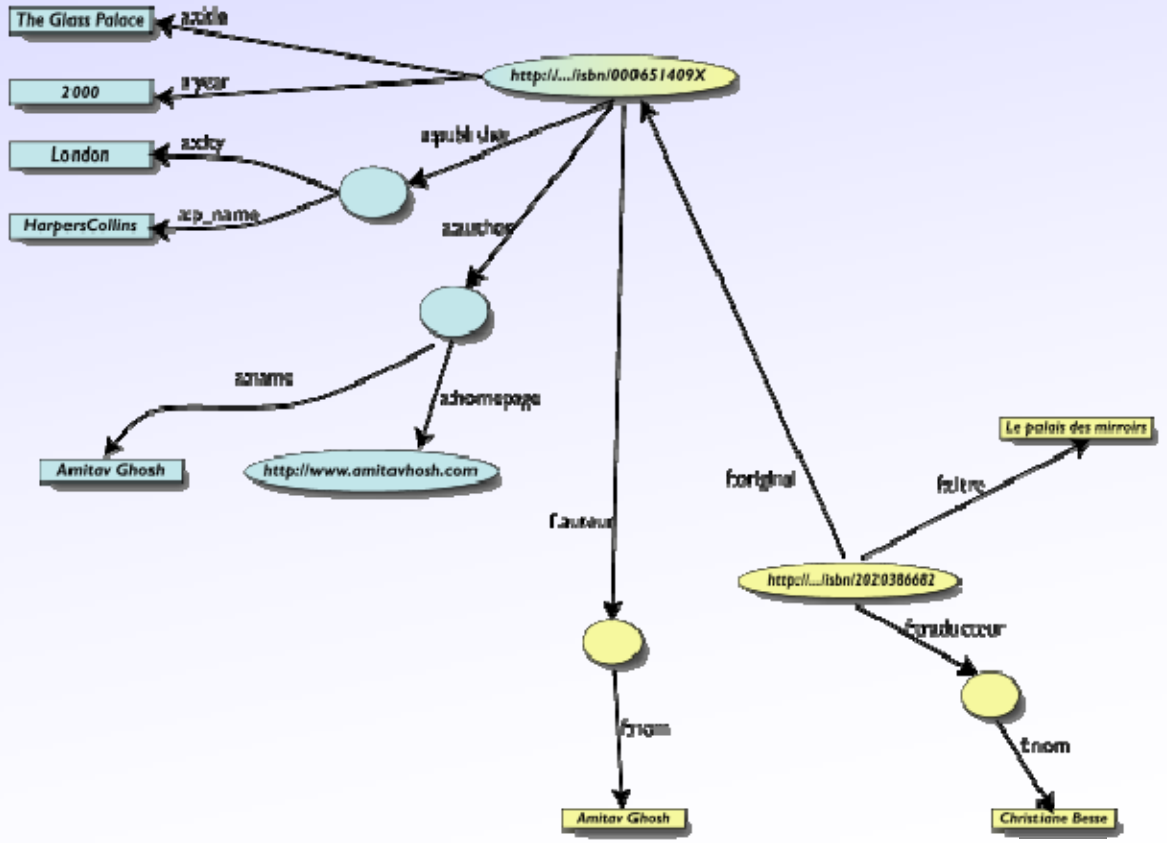


3rd: merge identical resources



Start making queries…

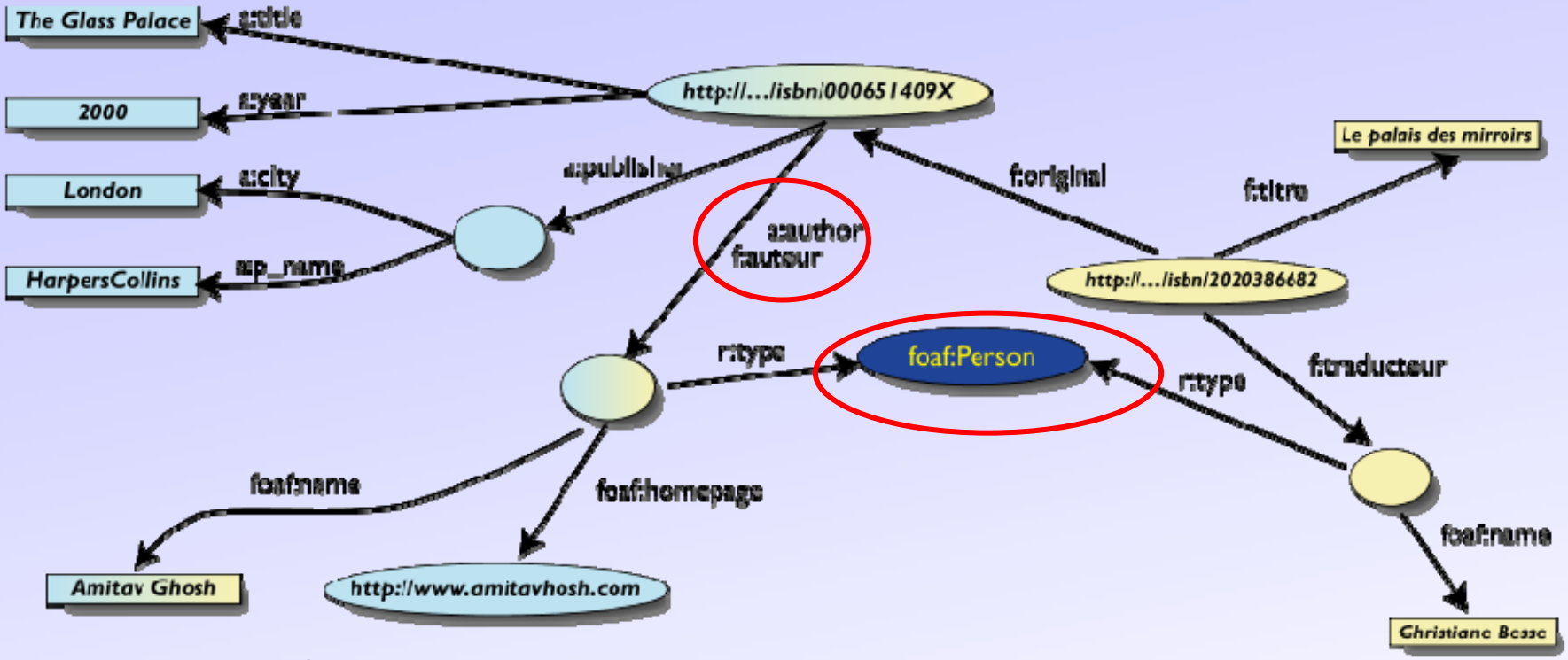
- User of data “F” can now ask queries like:
  - “give me the title of the original”
- This information is not in the dataset “F”…
- …but can be retrieved by merging with dataset “A”!



However, more can be achieved…

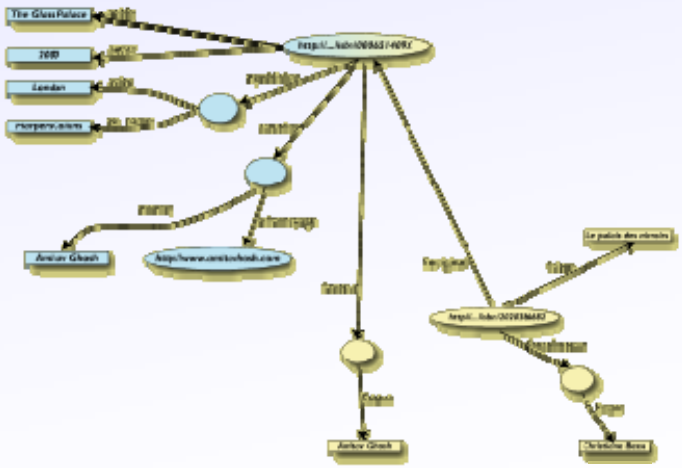
- We “feel” that **`a:author`** and **`f:auteur`** should be the same
- But an automatic merge does not know that!

3rd revisited: use the extra knowledge



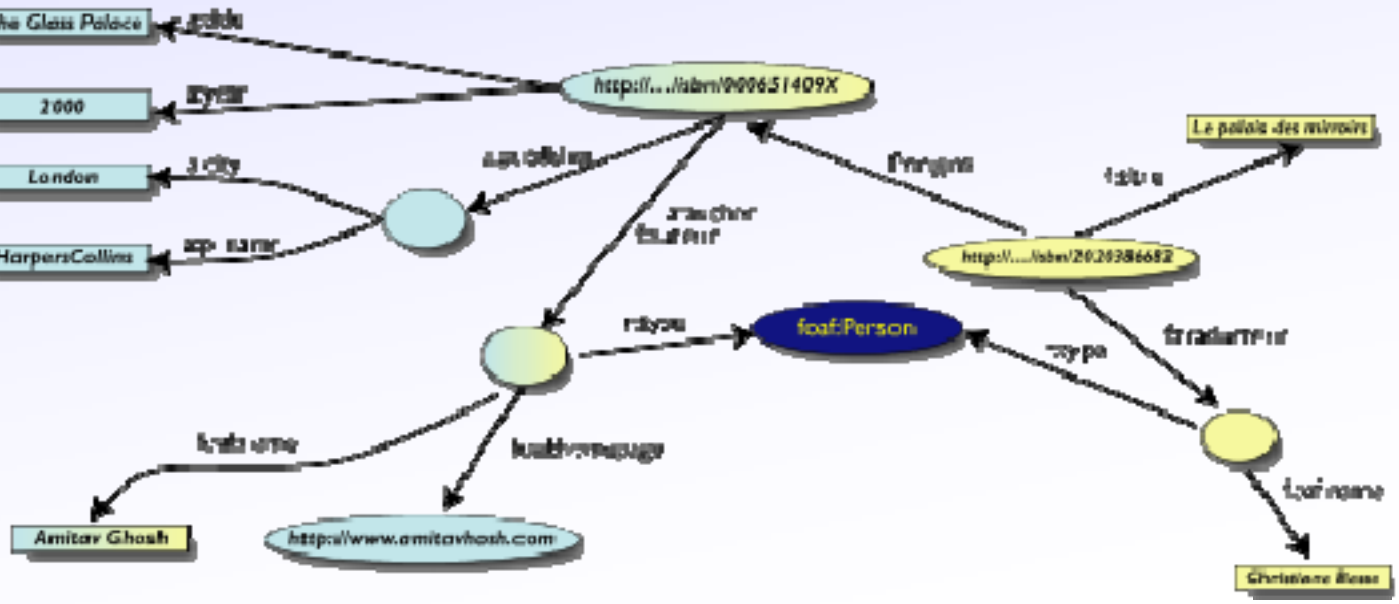
Add some extra information to the merged data:

1) **`a:author`** same as **`f:auteur`**
2) both identify a "**Person**", a term that a community (foaf:Person) may have already defined:
   - a "Person" is uniquely identified by his/her name and, say, homepage
   - it can be used as a "category" for certain type of resources



Start making richer queries!

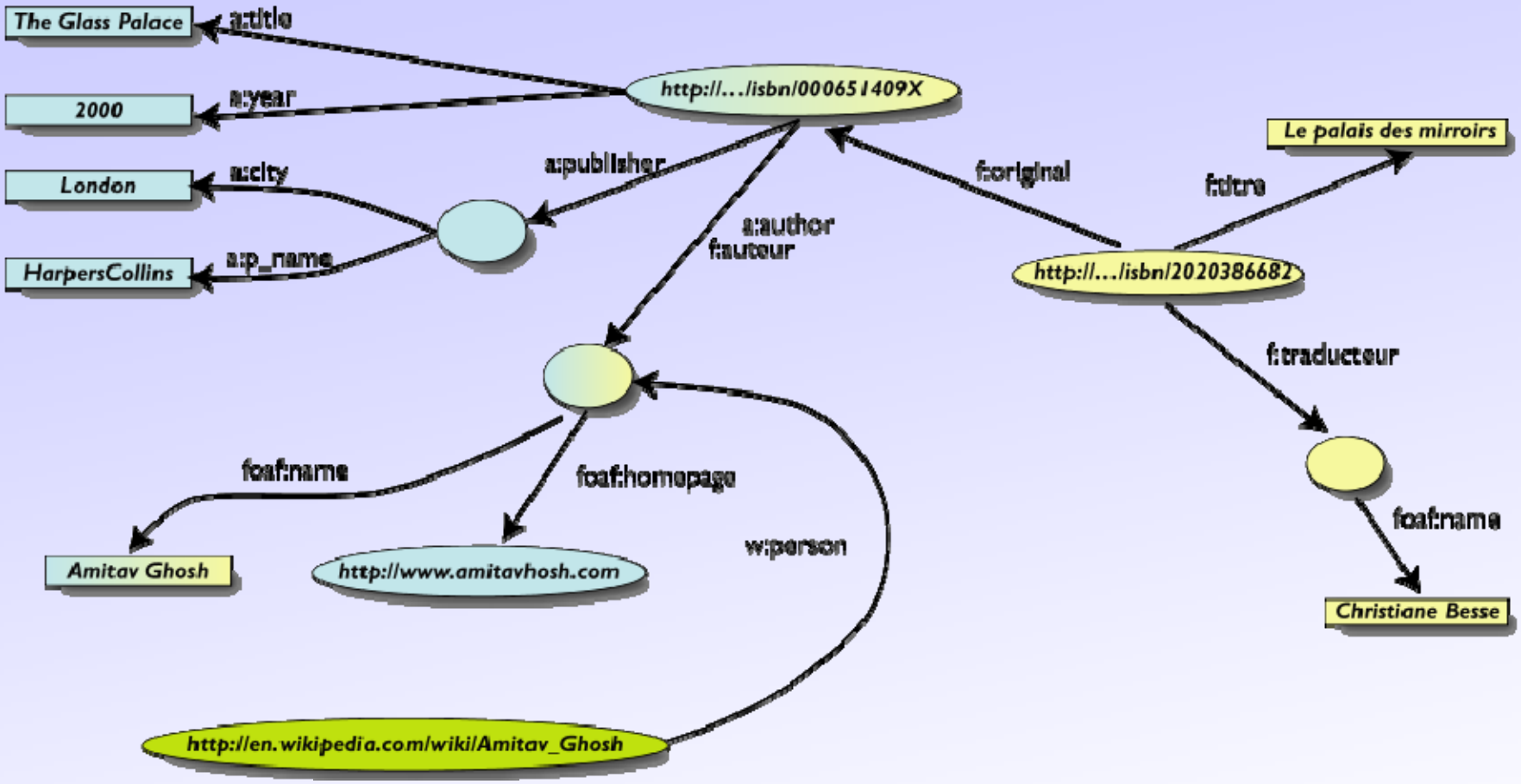
- User of dataset “F” can now query:
  - “give me the home page of the original’s author”
- The information is not in datasets “F” or “A”…
- …but was made available by:
  - merging datasets “A” and datasets “F”
  - adding three simple extra statements as an extra “glue”



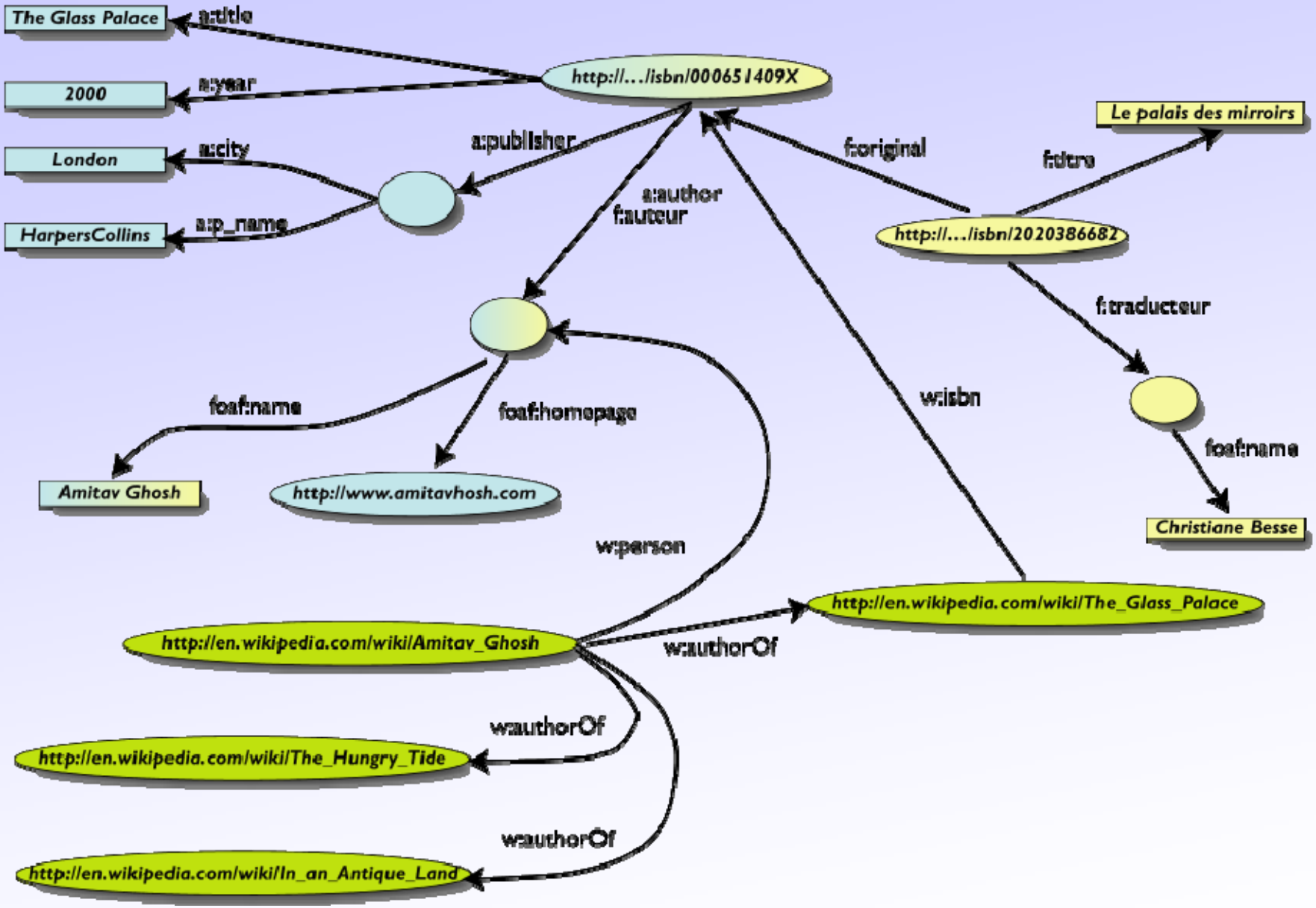
Combine with different datasets

- Via, e.g., the “Person”, the dataset can be combined with other sources
- For example, data in **Wikipedia** can be extracted using dedicated tools

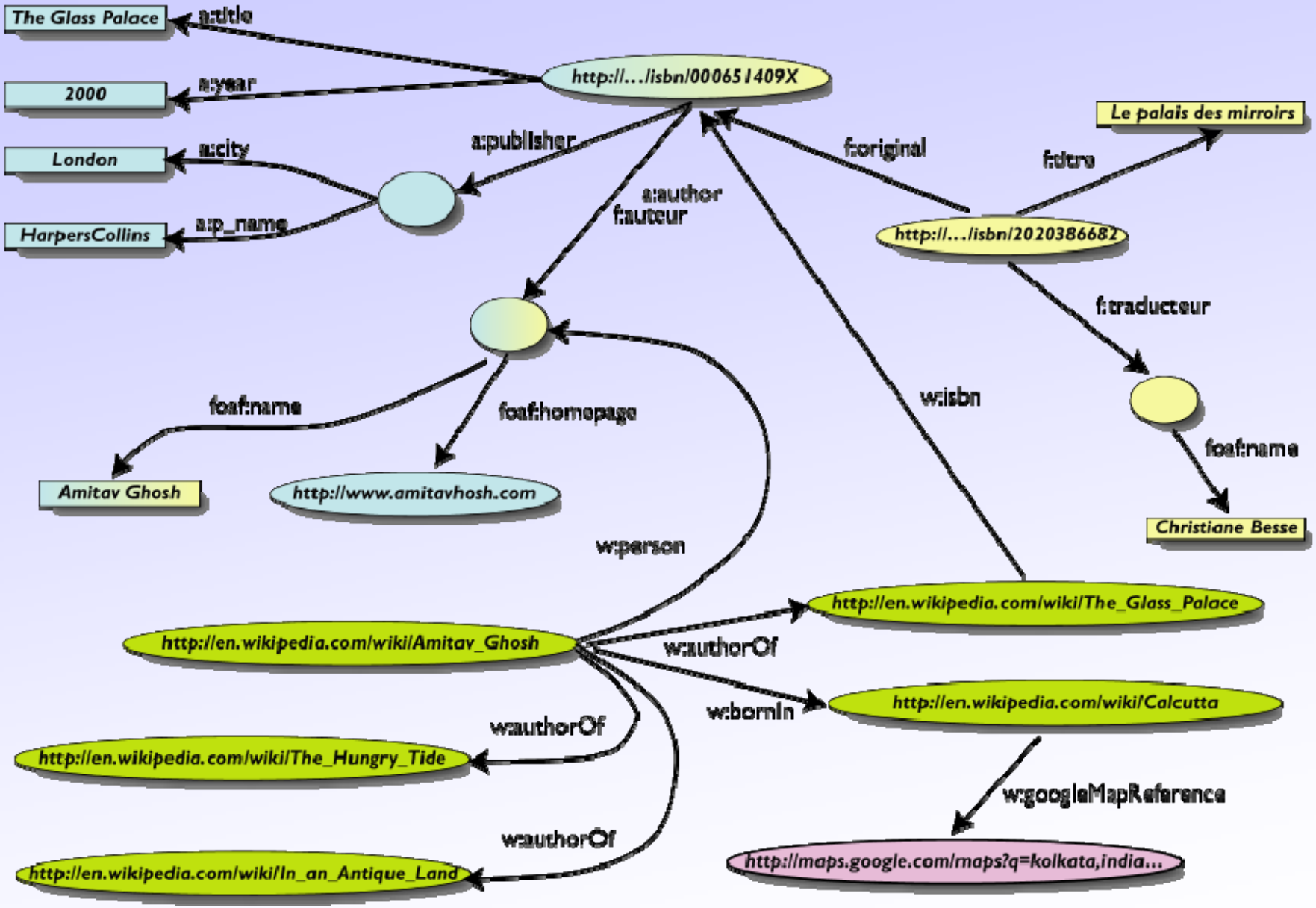
Merge with Wikipedia data



Merge with Wikipedia data



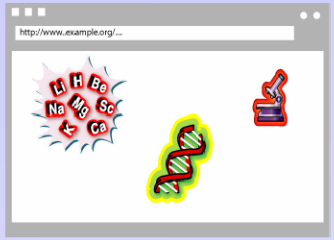
Merge with Wikipedia data

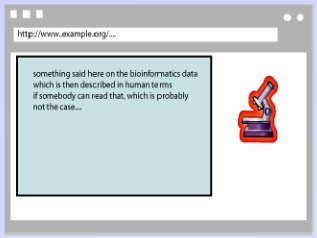


Is that surprising?

- It may look like it but, in fact, it should not be…
- What happened via automatic means is done every day by Web users!
- The difference: a bit of extra rigour so that machines could do this, too

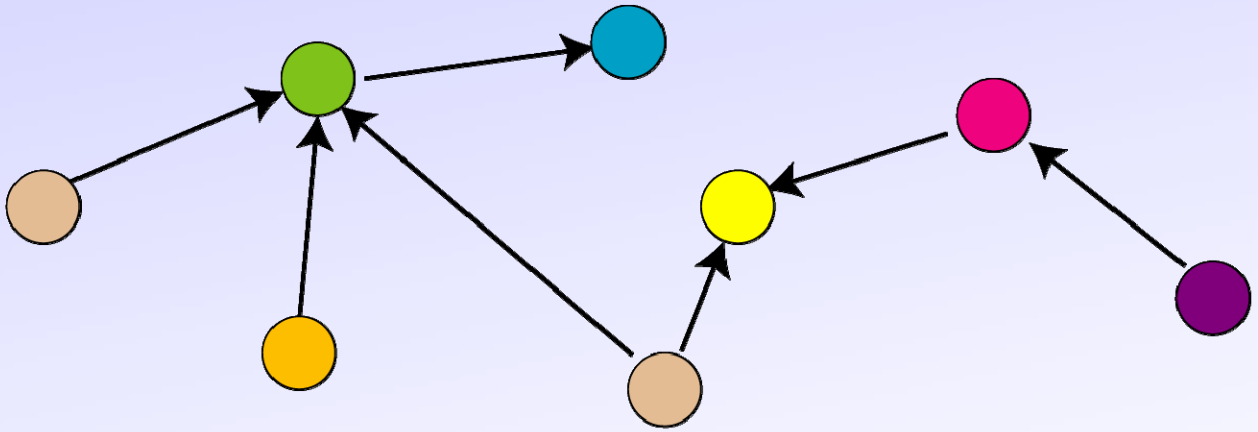
What did we do?





Applications

Query, Manipulate, etc.



Data represented in abstract format

Map, Expose, etc.

Data in various formats

It could become even more powerful

- We could add extra knowledge to the merged datasets
  - e.g., a full classification of various types of library data
  - geographical information
  - etc.
- This is where ***ontologies*** come in
  - ontologies can be relatively simple and small, or huge, or anything in between…
- Even more powerful queries can be asked as a result

The abstraction pays off because…

- … the graph representation is independent of the exact structures
- … a change in local database schema's, XHTML structures, etc, do not affect the whole
  - "schema independence"
- … new data, new connections can be added seamlessly

The network effect

- Through URI-s we can link any data to any data
- The "network effect" is extended to the (Web) data
- "Mashup on steroids" become possible

So where is the Semantic Web?

- The Semantic Web provides technologies to make such integration possible!
- Hopefully you get a full picture at the end of the tutorial…

The Basis: RDF

RDF triples

- Let us begin to formalize what we did!
  - we “connected” the data…
  - but a simple connection is not enough… data should be named somehow
  - hence the **RDF Triples:** ***a labelled connection between two resources***

RDF triples (cont.)

- An RDF Triple **(s,p,o)** is such that:
  - "**s**", "**p**" are URI-s, ie, resources on the Web; "**o**" is a URI or a literal
    - "**s**", "**p**", and "**o**" stand for :

      "**subject**", "**property**" or "**predicate**", and "**object**"
  - here is the complete triple:

```
(<http://…isbn…6682>, <http://…/original>, <http://…isbn…409X>)
```

- *RDF* is a general model for such triples (with machine readable formats like RDF/XML, Turtle, N3, RXR, …)
- RDF triples are also referred to as "triplets", or "statements"

RDF triples (cont.)

- Resources can use *any* URI;
  - it can denote an element within an XML file on the Web, not only a “full” resource, e.g.:
    - `http://www.example.org/file.xml#element(home)`
    - `http://www.example.org/file.html#home`
    - `http://www.example.org/file2.xml#xpath1(//q[@a=b])`
- RDF triples form a directed, labelled graph (the best way to think about them!)

RDF/XML Document example

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="http://www.w3schools.com/rdf/">

<rdf:Description rdf:about="http://www.w3schools.com">
  <si:title>W3Schools</si:title>
  <si:author>Jan Egil Refsnes</si:author>
</rdf:Description>

</rdf:RDF>
```

<rdf:RDF> is the **root element of an RDF document**.

- defines the XML document to be an RDF document.
- contains a reference to the RDF namespace.

XML Elements

- Τα «πράγματα» στα οποία αναφέρεται το XML κείμενο
  - Π.χ. Βιβλία, συγγραφείς, εκδότες
- Ένα element αποτελείται από:
  - an opening tag
  - the content
  - a closing tag

```
<lecturer>David Billington</lecturer>
```

XML Elements (2)

- Υπάρχει σχεδόν πλήρης ελευθερία επιλογής Tag names
- Ο πρώτος χαρακτήρας πρέπει να είναι γράμμα, underscore, ή colon
- Κανένα όνομα δε πρέπει να ξεκινά με το λεκτικό “xml” ή συνδυασμούς αυτού
  - Π.χ. “Xml”, “xML”

Περιεχόμενο XML Elements

- Το περιεχόμενο μπορεί να είναι κείμενο, άλλα elements, ή τίποτα

```
<lecturer>
 <name>David Billington</name>
 <phone> +61 – 7 – 3875 507 </phone>
</lecturer>
```

- Στην XML κάποιοι χαρακτήρες είναι δεσμευμένοι και δεν μπορούν να τοποθετηθούν στο περιεχόμενο ενός κειμένου παρά μόνον με ειδικό τρόπο:
  - ο χαρακτήρας & μπορεί να εισαχθεί ως &
  - ο χαρακτήρας < μπορεί να εισαχθεί ως <
  - ο χαρακτήρας > μπορεί να εισαχθεί ως >
  - ο χαρακτήρας “ μπορεί να εισαχθεί ως "
  - ο χαρακτήρας ‘ μπορεί να εισαχθεί ως '

XML Attributes

- Το attribute είναι ένα ζεύγος name-value pair μέσα στο opening tag ενός element

```
<lecturer name="David Billington" phone="+61 – 7 – 3875 507"/>
```

Παράδειγμα XML Attribute

```
<order orderNo="23456" customer="John Smith"
   date="October 15, 2002">
   <item itemNo="a528" quantity="1"/>
   <item itemNo="c817" quantity="3"/>
</order>
```

Το ίδιο παράδειγμα χωρίς Attributes

```
<order>
  <orderNo>23456</orderNo>
  <customer>John Smith</customer>
  <date>October 15, 2002</date>
  <item>
   <itemNo>a528</itemNo>
   <quantity>1</quantity>
  </item>
  <item>
   <itemNo>c817</itemNo>
   <quantity>3</quantity>
   </item>
</order>
```

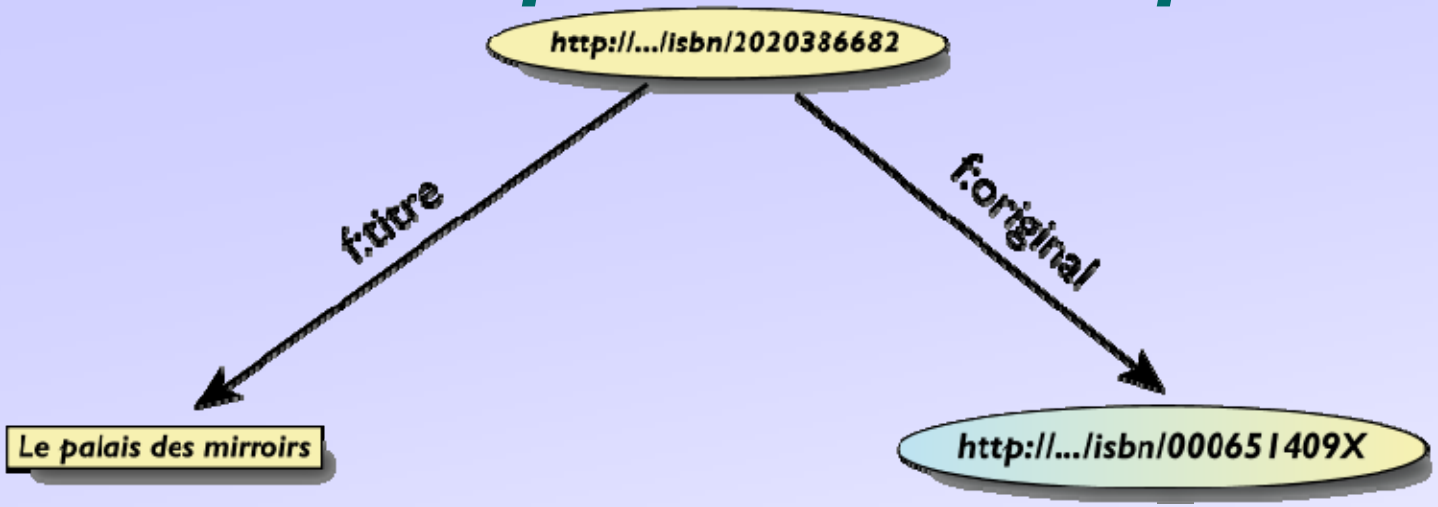
XML Elements vs Attributes

- Τα Attributes μπορούν να υποκατασταθούν από elements
- Πότε χρησιμοποιεί κανείς attributes ή elements είναι θέμα επιλογής
- Υπάρχουν οι εξής περιορισμοί
  - Εαν υπάρχει περίπτωση να έχουμε δύο ή περισσότερες τιμές για το συγκεκριμένο είδος πληροφορίας, χρησιμοποιούμε element

```
<book>
   <author>John Mavropoulos</author>
   <author>Jim Pavlopoulos</author>
</book>
```

  - Τα attributes **δεν μπορούν** να είναι nested

A simple RDF example



In XML/RDF :

```
<rdf:Description rdf:about="http://…/isbn/2020386682">
    <f:titre xml:lang="fr">Le palais des mirroirs</f:titre>
    <f:original rdf:resource="http://…/isbn/000651409X"/>
</rdf:Description>
```

In Turtle :

```
<http://…/isbn/2020386682>
    f:titre "Le palais des mirroirs"@fr ;
    f:original <http://…/isbn/000651409X> .
```

(Note: namespaces are used to simplify the URI-s)

URI-s play a fundamental role

- URI-s made the merge possible
- *URI-s ground RDF into the Web*
  - information can be retrieved using existing tools
  - this makes the “Semantic Web”, well… “Semantic *Web*”

RDF in programming practice

- For example, using Java+Jena (HP's Bristol Lab):
  - a "Model" object is created
  - the RDF file is parsed and results stored in the Model
  - the Model offers methods to retrieve:
    - Triples,
    - (property,object) pairs for a specific subject
    - (subject,property) pairs for specific object
- Similar tools exist in Python, PHP, etc.

Jena Example

```
// create a model
  Model model=new ModelMem();
  Resource subject=model.createResource("URI_of_Subject")
  // 'in' refers to the input file
  model.read(new InputStreamReader(in));
  StmtIterator iter=model.listStatements(subject,null,null);
  while(iter.hasNext()) {
     st = iter.next();
     p = st.getProperty();
     o = st.getObject();
     do_something(p,o);
  }
```

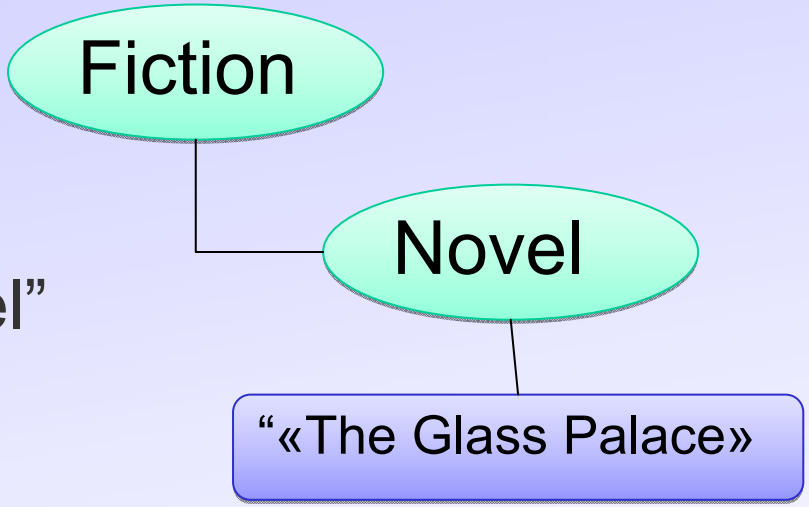
One level higher up
(RDFS, Datatypes)

Need for RDF schemas

- First step towards the "extra knowledge":
  - define the terms we can use
  - what **restrictions** apply
  - what extra **relationships** are there?
- Officially: "RDF Vocabulary Description Language"
  - the term "Schema" is retained for historical reasons…

Classes, resources, …

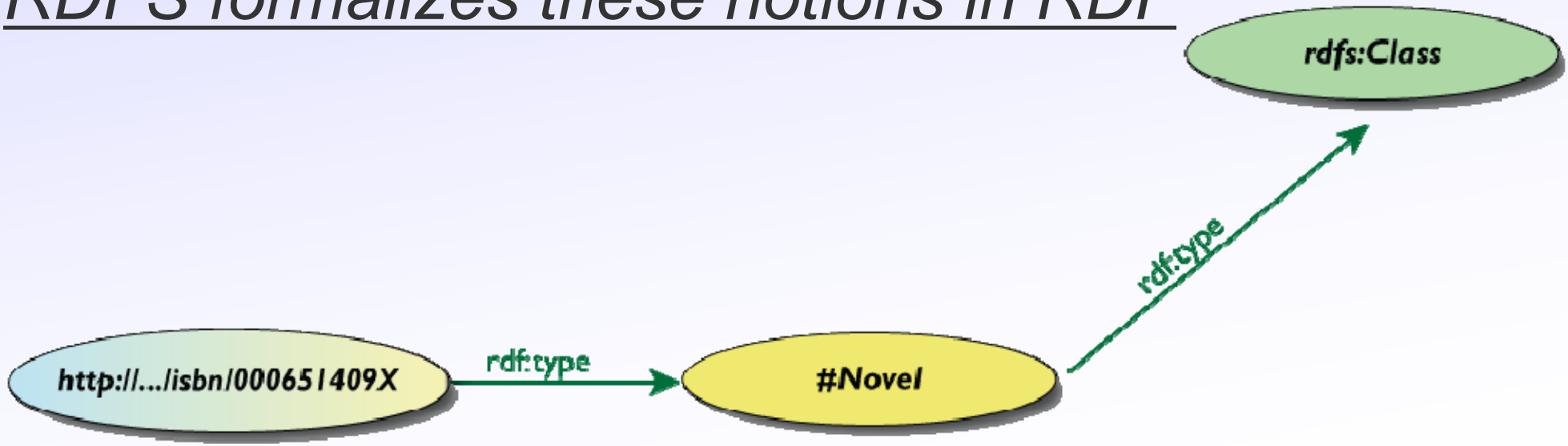
- Think of well known traditional ontologies or taxonomies:
  - use the term "**novel**"
  - "**every novel is a fiction**"
  - "«The Glass Palace» is a novel"
  - etc.



- RDFS defines **resources** and **classes**:
  - everything in RDF is a "resource"
  - "**classes**" are also resources, **but**…
  - …they are also a **collection of possible resources** (i.e., "individuals")
    - "fiction", "novel", …

Classes, resources, … (cont.)

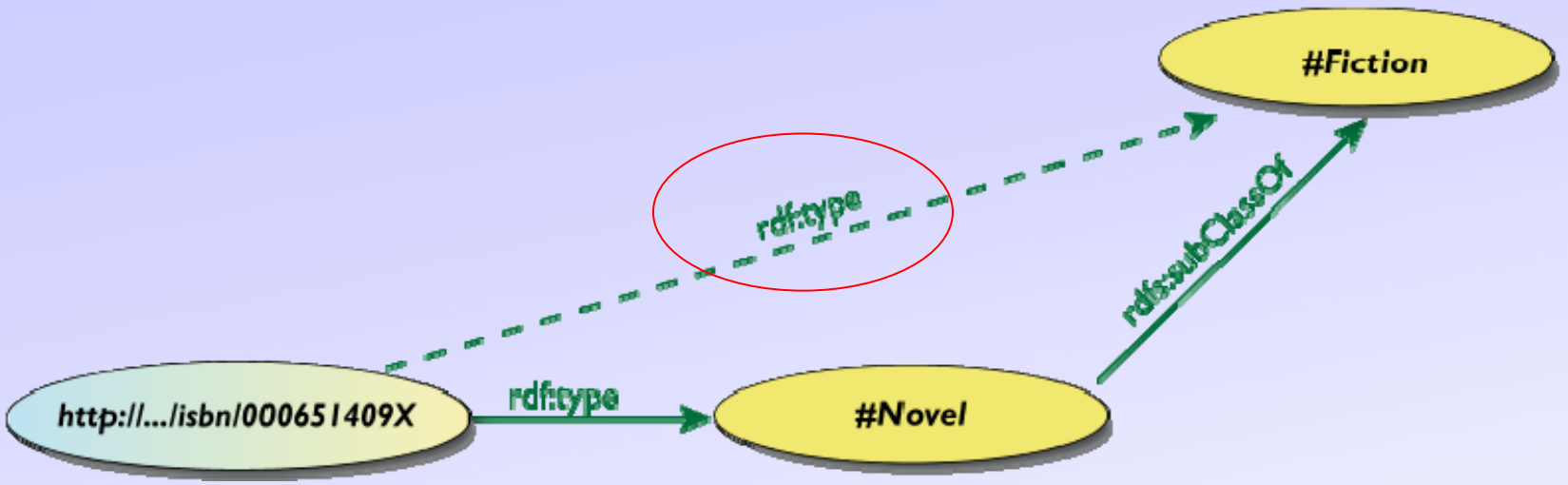
- **Relationships** are defined among classes/resources:
  1) "**typing**": an individual belongs to a specific class
     - "«The Glass Palace» is a novel"
     - to be more precise: "«http://.../**000651409X**» is a novel"
  2) "**subclassing**": *all* instances of one are also the instances of the other ("every novel is a fiction")
- *RDFS formalizes these notions in RDF*



Further remarks on types

- A resource may belong to several classes
  - rdf:type is just a property…
  - "«The Glass Palace» is a novel, but «The Glass Palace» is also an «inventory item»…"
  - i.e., it is *not* like a datatype!
- The type information may be very important for applications
  - e.g., it may be used for a categorization of possible nodes
  - probably the most frequently used RDF property…

Inferred properties




```
(<http://…/isbn/000651409X> rdf:type #Fiction)
```

- is not in the original RDF data…
- …but can be inferred from the RDFS rules
- RDFS environments return that triple, too

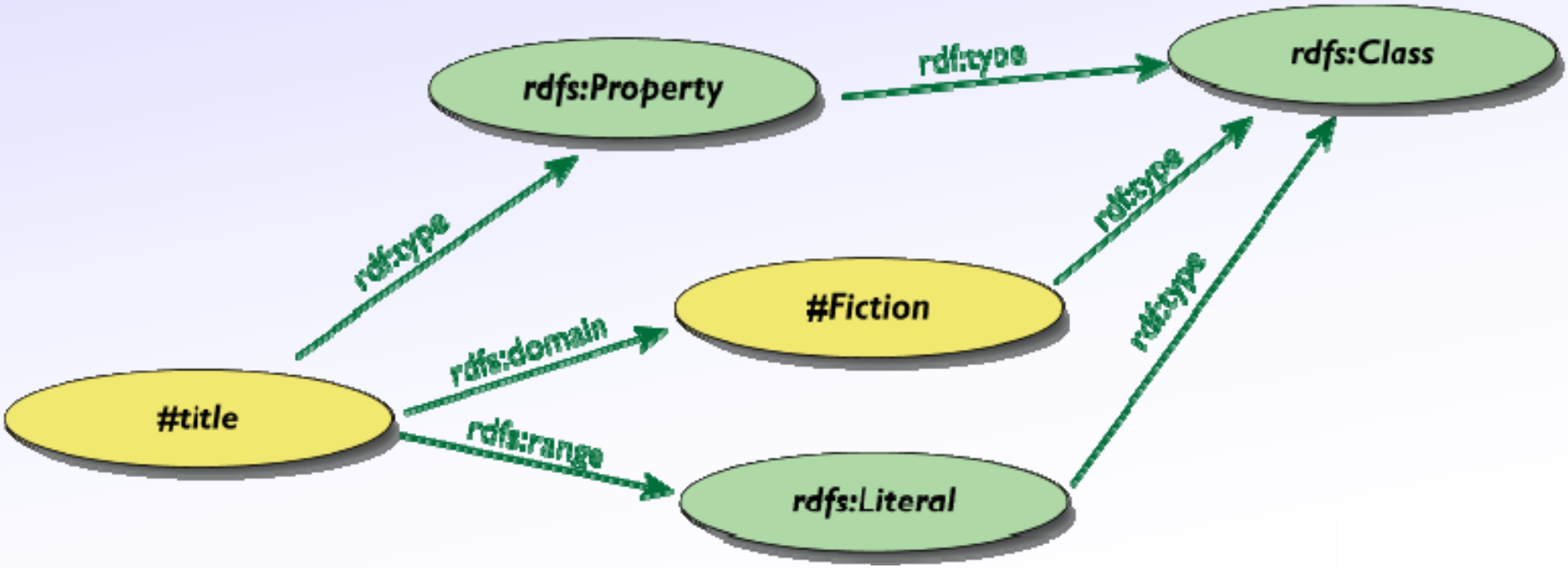
Inference: let us be formal…

- The RDF Semantics document has a list of (33) *entailment rules* :
  - "if such and such triples are in the graph, add this and this"
  - do that recursively until the graph does not change
  - See: http://www.w3.org/TR/rdf-mt/#RDFSRules
- The relevant rule for our example:

```
If:
   uuu rdfs:subClassOf xxx .
   vvv rdf:type uuu .
Then add:
   vvv rdf:type xxx .
```

Properties

- Property is a special class (**`rdf:Property`**)
  - properties are also resources identified by URI-s
- There is also a possibility for a "sub-property"
  - all resources bound by the "sub" are also bound by the other
- Range and domain of properties can be specified
  - i.e., *what type of resources serve as object and subject*



New relations can be deduced

- If we declare that a Property “title” such as:

```
:title
  rdf:type     rdf:Property;
  rdfs:domain  :Fiction;
  rdfs:range   rdfs:Literal.

- And an RDF triple stating that :

<http://…/isbn/000651409X> :title "The Glass Palace" .
```

- then the system can *infer* that:

```
<http://…/isbn/000651409X> rdf:type :Fiction .
```

Το domain του ***:title*** *έχει δηλωθεί (με RDFS)* ***:Fiction*** *=> οποιοδήποτε resource συνδέται με το property :title (από αριστερά) με κάποιο άλλο, πρέπει να ανήκει στο domain :Fiction*

Literals

- **Objects may be literals** (not URI's)
- Literals may have a data type
  - floats, integers, booleans, etc, **defined in XML Schemas**
  - **OR** full **XML fragments**
- (Natural) language can also be specified

Examples for datatypes

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <page_number rdf:datatype="http://...#integer>543</page_number>
   <publ_date rdf:datatype="http://...#gYear>2000</publ_date>
   <price rdf:datatype="http://...#float>6.99</price>
</rdf:Description>
```

Examples for language tags

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <title xml:lang="en">The Glass Palace</title>
   <fr:titre xml:lang="fr">Le palais des mirroirs</fr:titre>
</rdf:Description>
```

XML literals in RDF/XML

- XML Literals
  - makes it possible to “include” XML vocabularies into RDF:

```
<rdf:Description rdf:about="#Path">
   <axsvg:algorithmUsed rdf:parseType="Literal">
      <math xmlns="...">
        <apply>
          <laplacian/>
          <ci>f</ci>
        </apply>
      </math>
   </axsvg:algorithmUsed>
</rdf:Description/>
```

Query RDF Data
(SPARQL)

RDF data access

- How do I *query* the RDF data?

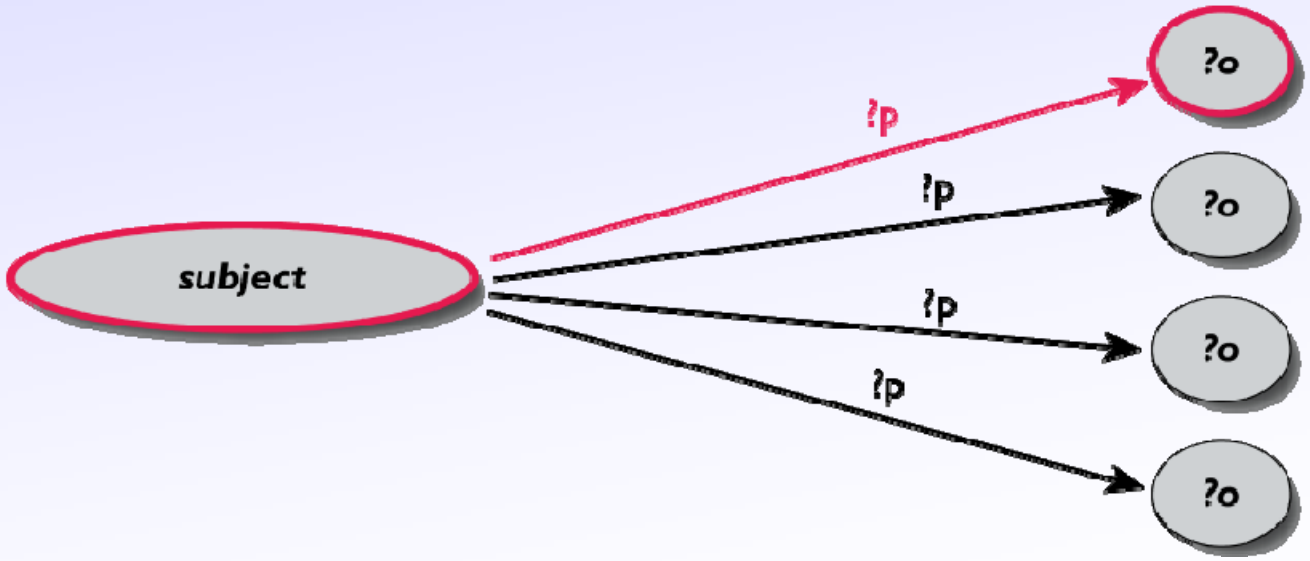
Querying RDF graphs

- Complex queries into the RDF data are necessary
  - something like: “give me the (a,b) pair of resources, for which there is an x such that (x parent a) and (b brother x) holds” (ie, return the uncles)
  - these rules may become quite complex
- The goal of SPARQL (Query Language for RDF)

Example in SPARQL

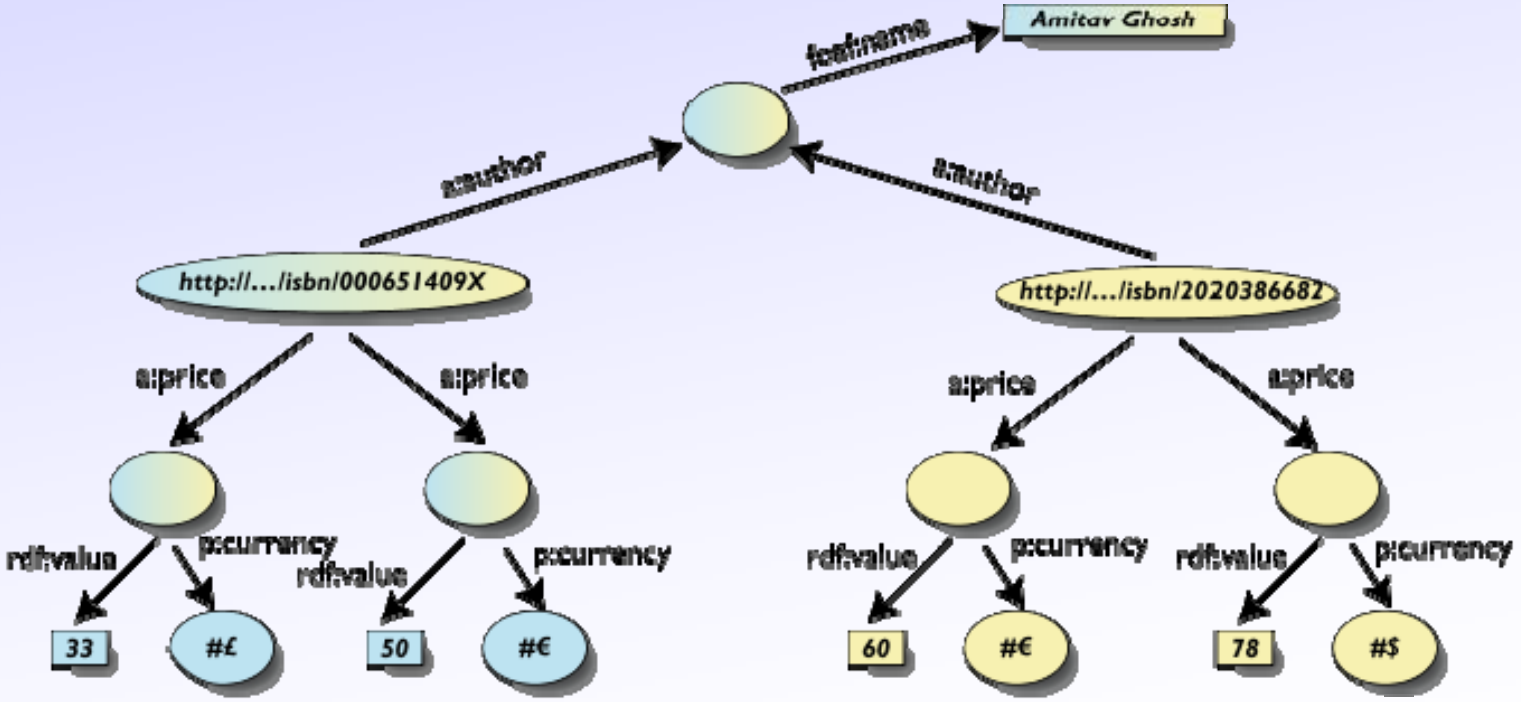
```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

- The triples in **WHERE** define the graph pattern, with **?p** and **?o** “unbound” symbols
- The query returns *all* **p**,**o** pairs



Simple SPARQL example

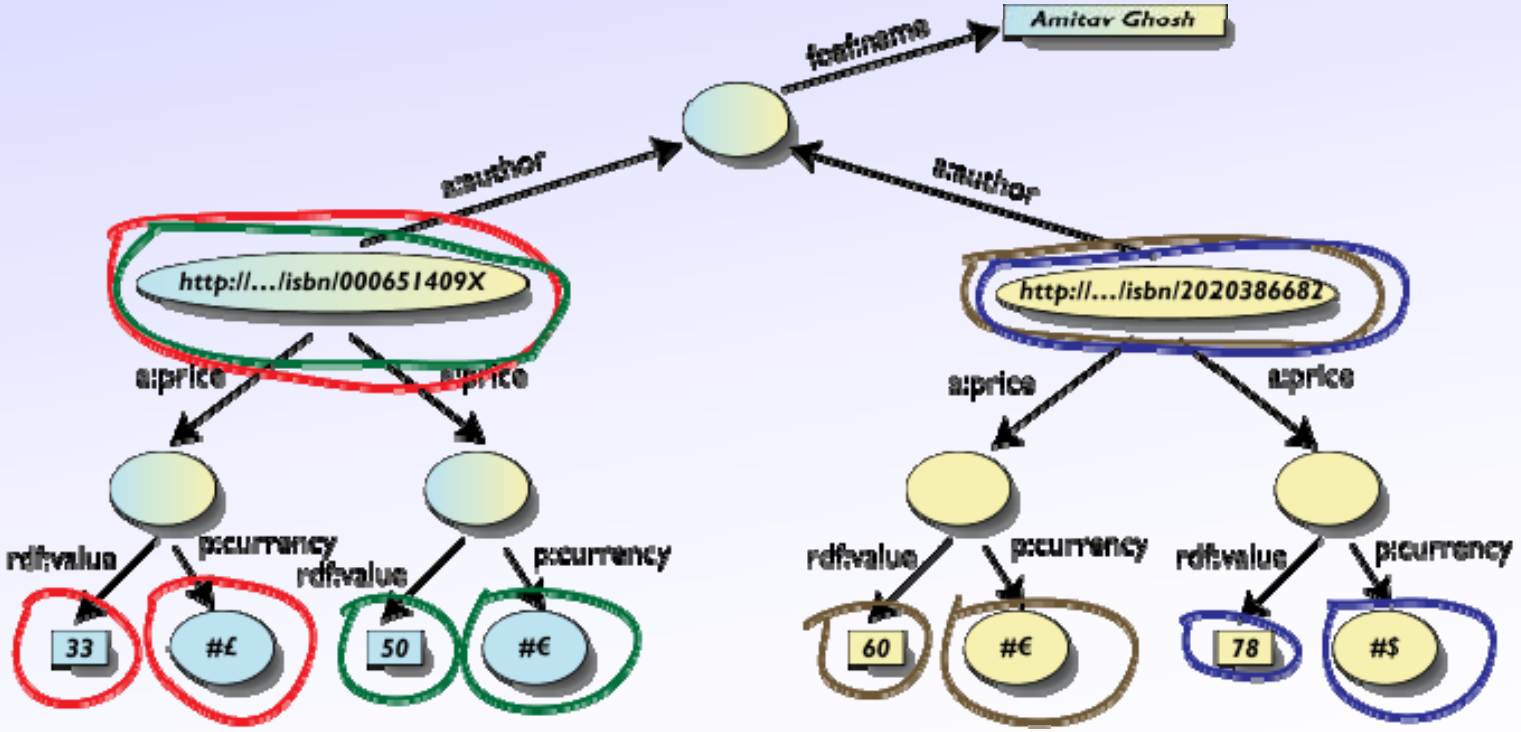
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

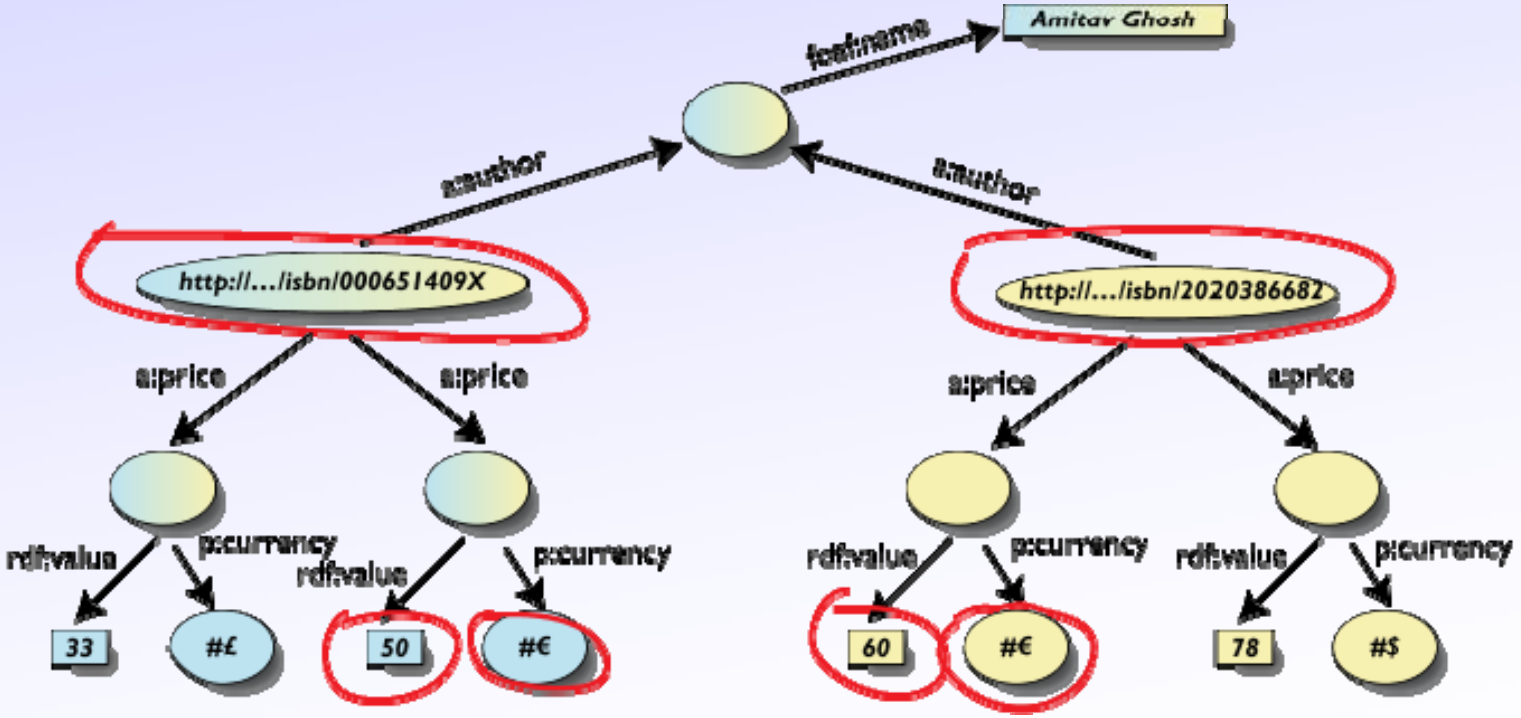
- Returns:
  [[<..49X>,33,£], [<..49X>,50,€], [<..6682>,60,€], [<..6682>,78,$]]



Pattern constraints

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        FILTER(?currency == € }
```

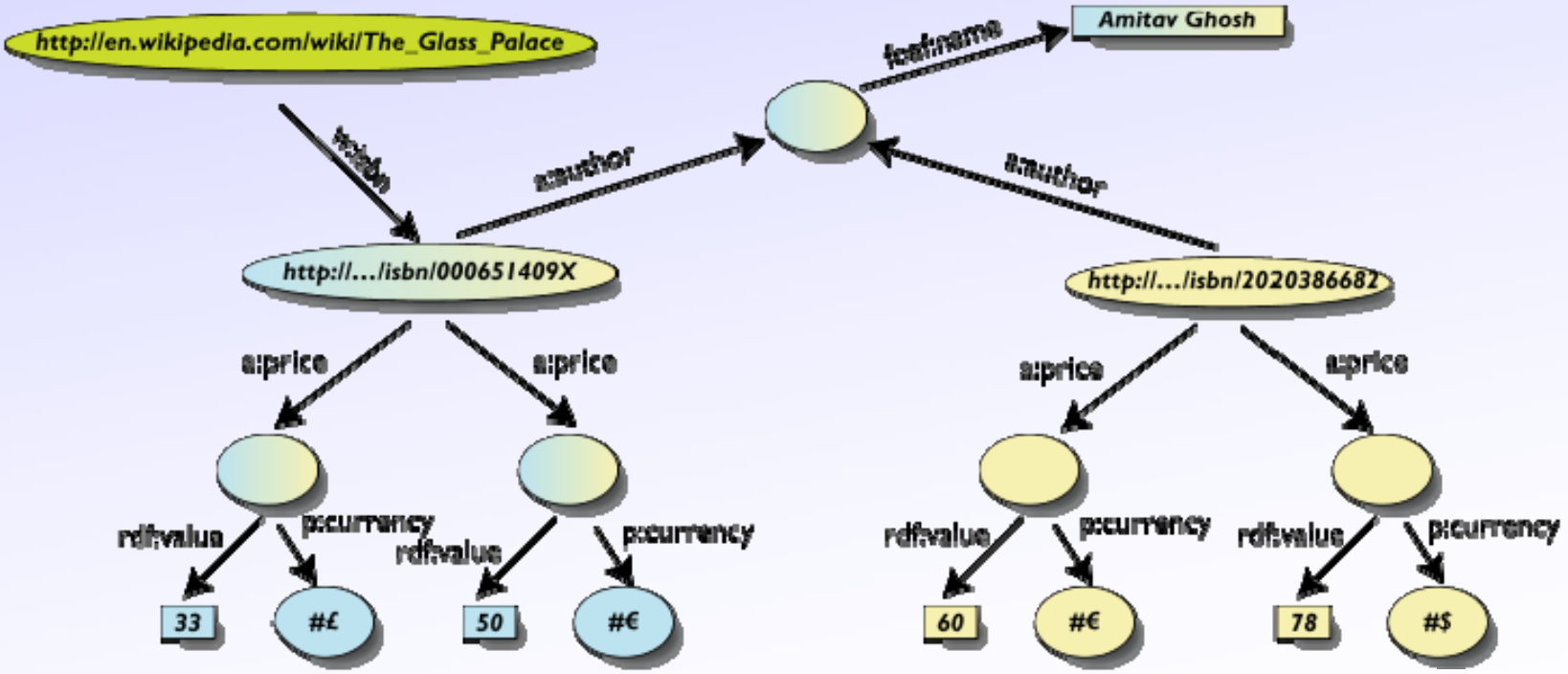
- Returns: [[<..409X>,50,€], [<..6682>,60,€]]



Optional pattern

```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

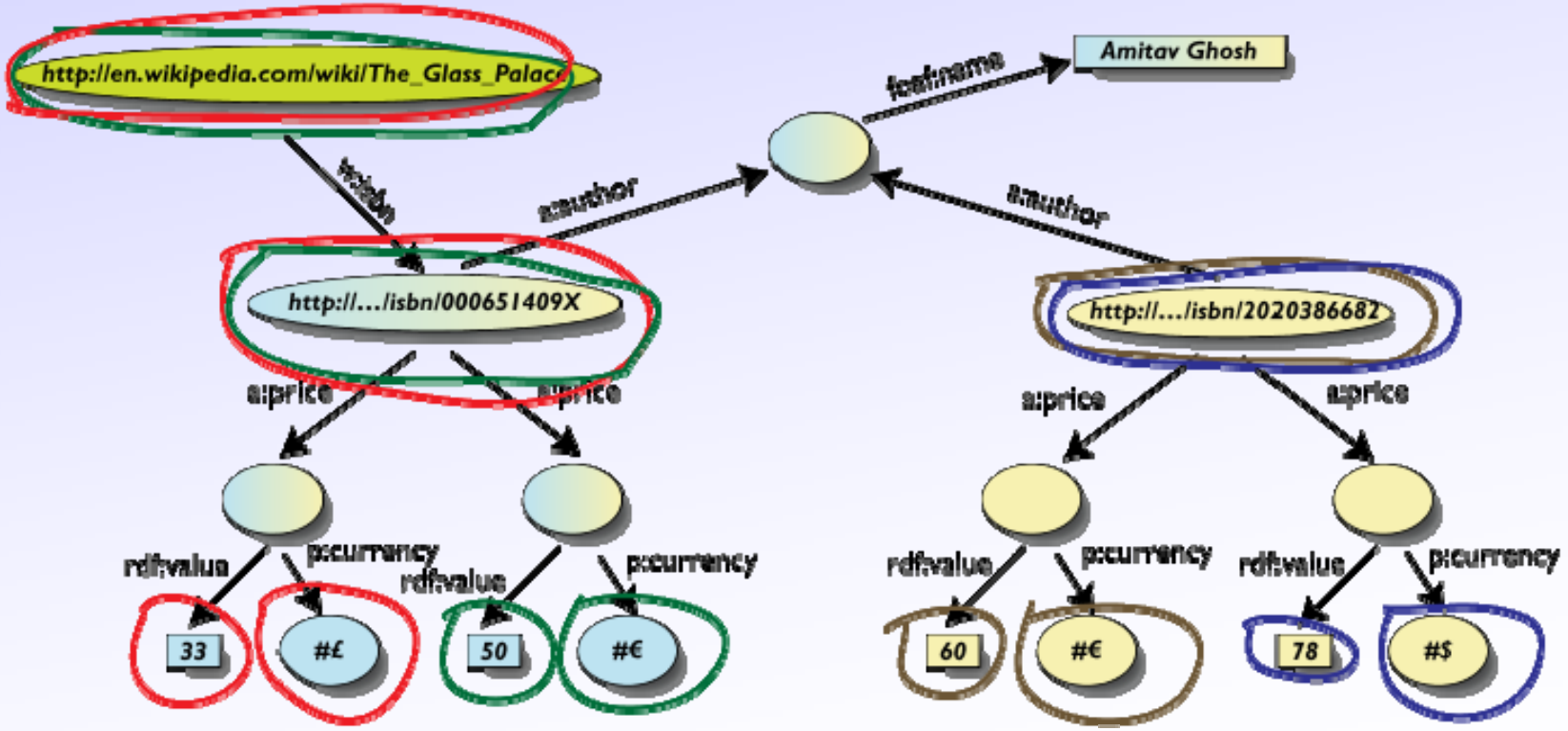
- Returns: [[<..49X>,33,£,<…Palace>], … , [<..6682>,78,$,]]



Optional pattern

```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

- Returns: [[<..49X>,33,£,<…Palace>], … , [<..6682>,78,$,]]



Other SPARQL features

- Limit the number of returned results; remove duplicates, sort them, …
- Specify several data sources (via URI-s) within the query (essentially, a merge!)
- *Construct* a graph combining a separate pattern and the query results
- Use datatypes and/or language tags when matching a pattern

Ontologies
(OWL)

Ontologies

- RDFS is useful, but does not solve all possible requirements
- Complex applications may want more possibilities

Limitations of the Expressive Power of RDF Schema

- Local scope of properties
  - **rdfs:range** defines the range of a property (e.g. eats) **for all classes**
  - In RDF Schema we cannot declare **range restrictions that apply to some classes only**
    - E.g. we cannot say that cows eat only plants, while other animals may eat meat, too
- Disjointness of classes
  - Sometimes we wish to say that classes are **disjoint** (e.g. **male** and **female**)
- Boolean combinations of classes
  - Sometimes we wish to build new classes by combining other classes using union, intersection, and complement
  - E.g. **person** is the disjoint union of the classes **male** and **female**

Limitations of the Expressive Power of RDF Schema (2)

- Cardinality restrictions
  - E.g. a person has exactly two parents, a course is taught by at least one lecturer
- Special characteristics of properties
  - Transitive property (like “greater than”)
  - Unique property (like “is mother of”)
  - A property is the inverse of another property (like “eats” and “is eaten by”)

Ontologies (cont.)

- The term *ontologies* is used in this respect:

> "defines the concepts and relationships used to describe and represent an area of knowledge"

- Ie, there is a need for Web Ontology Language(s)
  - RDFS can be considered as a simple ontology language
- Languages should be a compromise between
  - rich semantics for meaningful applications
  - feasibility, implementability

Web Ontology Language = OWL

- OWL is an extra layer, a bit like RDF Schemas
  - own namespace, own terms
  - it relies on RDF Schemas
- It is a separate recommendation
  - actually… there is a 2004 version of OWL (“OWL 1”)
  - and there is an update (“OWL 2”) that has been finalized recently (2009)

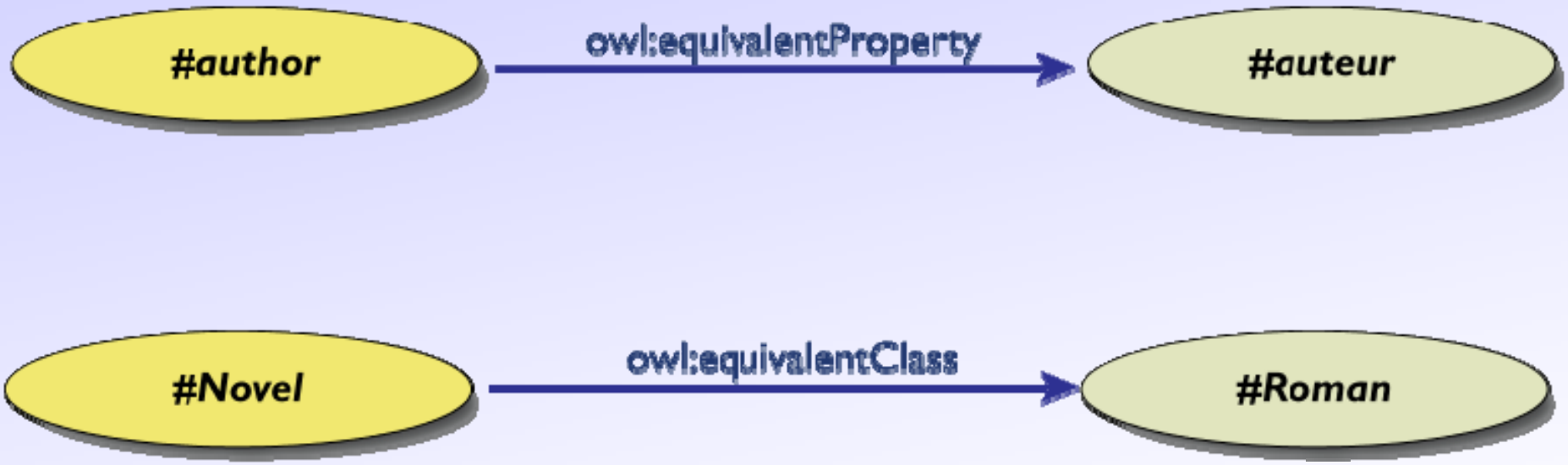
OWL is complex…

- OWL is a large set of additional terms
- We will not cover the whole thing here…

Term equivalences

- For classes:
  - **`owl:equivalentClass`**: two classes have the *same* individuals
  - **`owl:disjointWith`**: no individuals in common
- For properties:
  - **`owl:equivalentProperty`**
    - remember the **`a:author`** vs. **`f:auteur`**?
  - **`owl:propertyDisjointWith`**
- For individuals:
  - **`owl:sameAs`**: two URIs refer to the same concept (“individual”)
  - **`owl:differentFrom`**: negation of **`owl:sameAs`**

Other example: connecting to French



Typical usage of owl:sameAs

- Linking information about **“Amsterdam”** from one data set (**DBpedia**) to the other (**Geonames**):

```
<http://dbpedia.org/resource/Amsterdam>
  owl.sameAs <http://sws.geonames.org/2759793>;
```

- This is the main mechanism of “Linking” in the Linking Open Data project

Property characterization

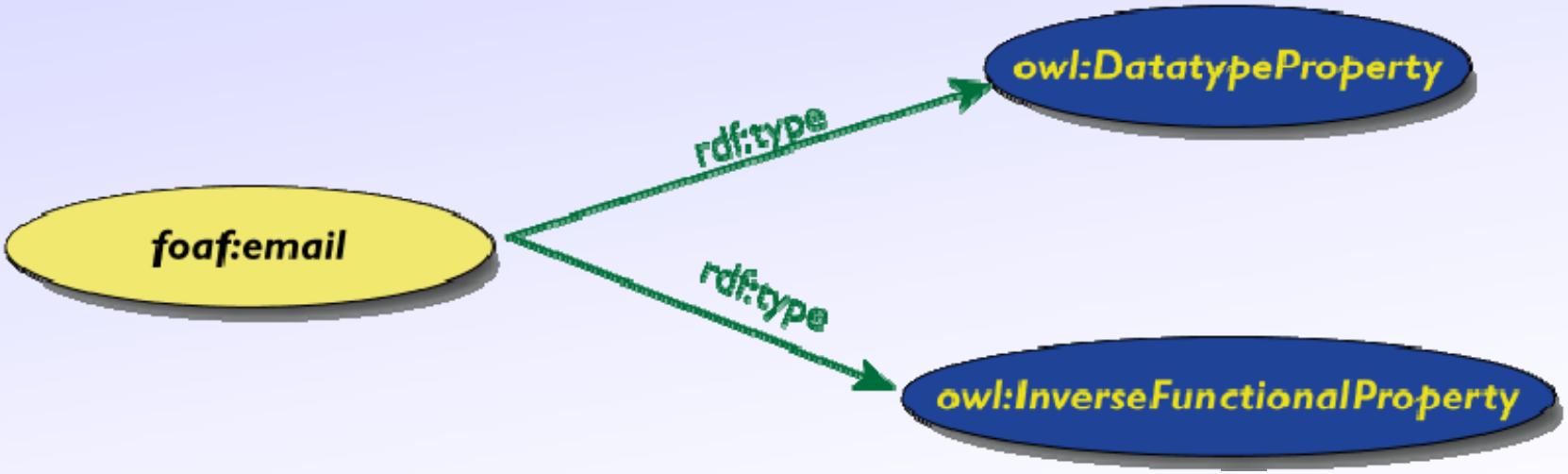
- In OWL, one can characterize the behaviour of properties
  - **symmetric,**
  - **transitive,**
  - **functional,**
  - **inverse functional**
  - …
- OWL also separates ***data*** and ***object*** **properties**
  - “datatype property” means that its range are typed literals

Property characterization

- OWL functional and inverse-functional properties indicate how many times a property can be used for a given subject or object
  - A functional property is one that has at most one value for any particular subject
    - An example is the hasBirthday relation between a person and his or her birthday
      - Everyone has just one birthday, so for any given subject (person), there can be just one object (birthday)
    - But, the owns relation between an owner and ownee is not functional. People can own more than one thing.
- Inverse functional properties do the same in reverse
  - For any object, there is only one subject for a particular inverse functional property
    - The has_ISBN relation is inverse functional
    - For any ISBN, there is only one book that has that ISBN
    - The has_ISBN relation may not be functional

Characterization example

- "**`foaf:email`**" may be defined as "inverse functional"
  - i.e., two different subjects cannot have identical objects



What this means is…

- If the following holds in our triples:

```
:email rdf:type owl:InverseFunctionalProperty.
<A> :email "mailto:a@b.c".
<B> :email "mailto:a@b.c".
```

then, processed through OWL, the following holds, too:

```
<A> owl:sameAs <B>.
```

- I.e., *new relationships* were discovered again (beyond what RDFS could do)

Other property characterizations

- There may be an inverse relationship among properties, eg:

```
<somebook> ex:author <somebody>.
ex:author owl:inverseOf ex:authorOf.
```

yields, in OWL:

```
<somebody> ex:authorOf <somebook>.
```

- In OWL 2 properties may also be characterized as reflexive or irreflexive

Keys (OWL 2)

"if two persons have the same emails *and* the same homepages then they are identical"

- Identification is based on the identical values of *two* properties
- The rule applies to persons only

Previous rule in OWL 2

```
:Person rdf:type owl:Class;
   owl:hasKey (:email :homepage) .
```

What it means is…

If:

```
<A> rdf:type :Person ;
   :email    "mailto:a@b.c";
   :homepage "http://www.ex.org".

<B> rdf:type :Person ;
   :email    "mailto:a@b.c";
   :homepage "http://www.ex.org".
```

then, processed through OWL 2, the following holds, too:

```
<A> owl:sameAs <B>.
```

Classes in OWL

- In RDFS, you can subclass existing classes… that's all
- In OWL, you can *construct* classes from existing ones:
  - enumerate its content
  - through intersection, union, complement
  - etc

Classes in OWL (cont)

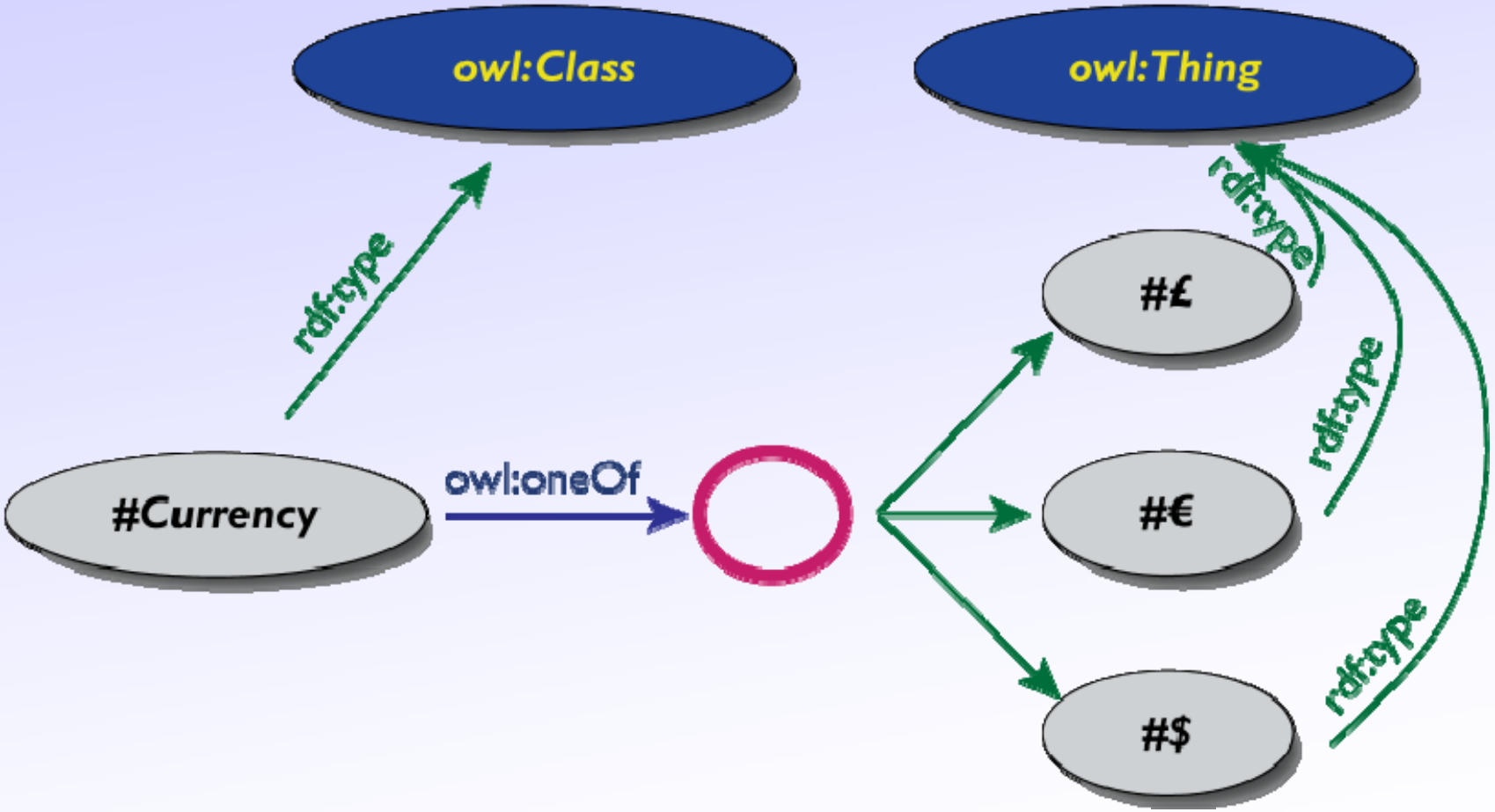
- OWL makes a stronger conceptual distinction between *classes* and *individuals*
  - there is a separate term for **`owl:Class`**, to make the difference
  - individuals are separated into a special class called **`owl:Thing`**
- Eg, a precise classification would be:

```
ex:Person rdf:type owl:Class.

<uri-for-Amitav-Ghosh>
    rdf:type owl:Thing;
    rdf:type owl:Person .
```

OWL classes can be “enumerated”

- The OWL solution, where possible content is explicitly listed:



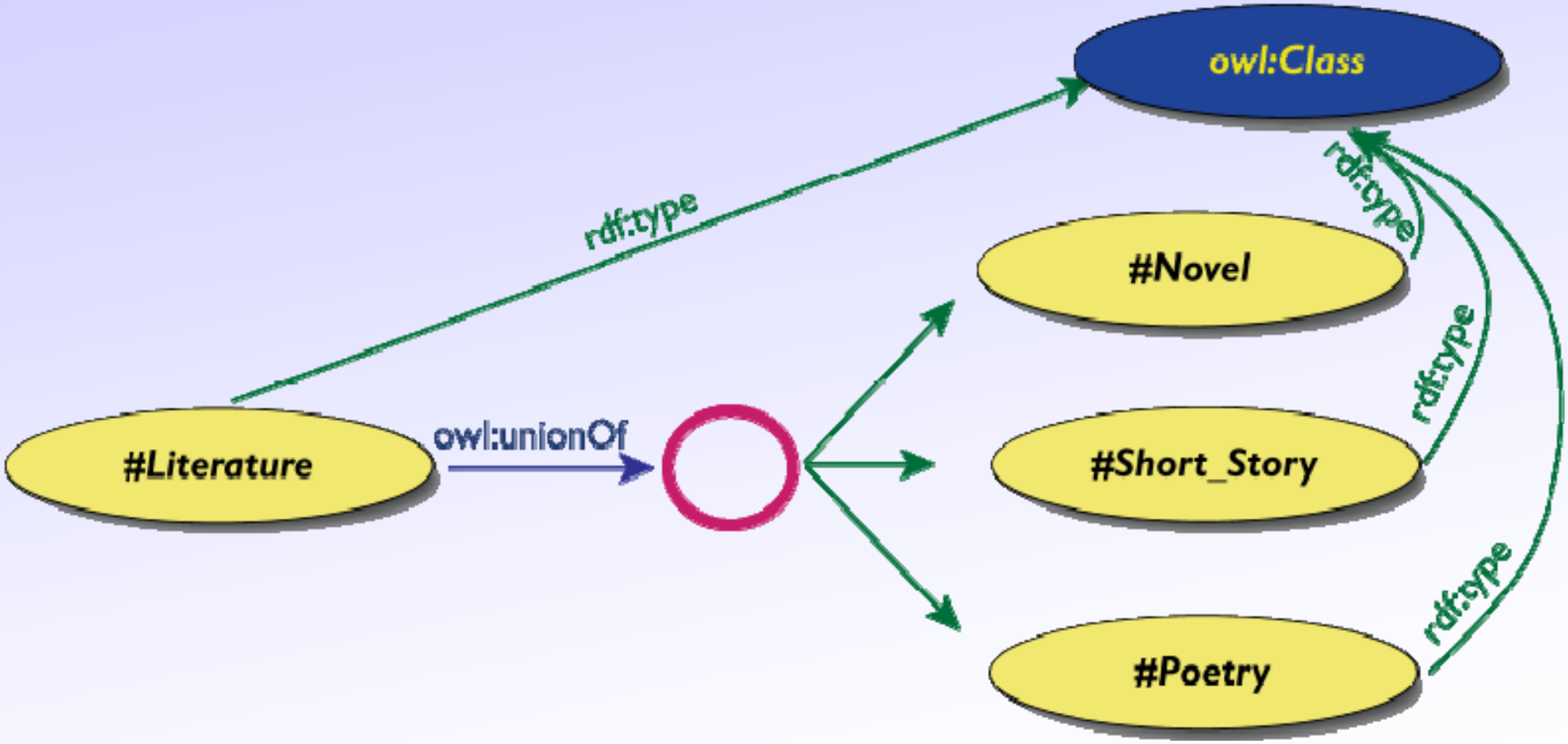
Same serialized

```
<owl:Class rdf:ID="Currency">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:ID="£"/>
    <owl:Thing rdf:ID="€"/>
    <owl:Thing rdf:ID="$"/>
    …
  </owl:oneOf>
</owl:Class>
```

- I.e., the class consists of *exactly* of those individuals

Union of classes

- Essentially, like a set-theoretical union:



Same serialized

```
<owl:Class rdf:ID="Literature">
   <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Novel"/>
       <owl:Class rdf:about="#Short_Story"/>
       <owl:Class rdf:about="#Poetry"/>
       …
   </owl:unionOf>
</owl:Class>
```

- Other possibilities: **`complementOf`**, **`intersectionOf`**, **…**

What we have so far…

- The OWL features listed so far are already fairly powerful
- E.g., various databases can be linked via **`owl:sameAs`**, functional or inverse functional properties, etc.
- Many inferred relationship can be found using a traditional rule engine

However… that may not be enough

- Very large vocabularies might require even more complex features
  - typical usage example: definition of all concepts in a health care environment
  - some major issues
    - the way classes (i.e., “concepts”) are defined
    - handling of datatypes
- OWL includes those extra features but… the inference engines become (much) more complex 

Property value restrictions

- Classes are created by *restricting* the property values on a (super)class
- For example: how would I characterize a “listed price”?
  - it is a price (which may be a general term),
    - but one that is given in **one of the “allowed” currencies** (say, €, £, or $)
  - more formally:
    - the value of “**`p:currency`**”, when applied to a resource on listed price, *must* take one of those values…
    - …thereby defining the class of “listed price”

Restrictions formally

- Defines a class of type **`owl:Restriction`** with a
  - reference to the property that is constrained
  - definition of the constraint itself
- One can, e.g., subclass from this node when defining a particular class

```
:Listed_Price rdfs:subClassOf [
      rdf:type              owl:Restriction;
      owl:onProperty        p:currency;
      owl:allValuesFrom     :Currency.
    ].
```

Possible usage…

If:

```
:Listed_Price rdfs:subClassOf [
      rdf:type              owl:Restriction;
      owl:onProperty        p:currency;
      owl:allValuesFrom     :Currency.
    ].

:Price rdf:type :Listed_Price .

:Price p:currency <someCurrency> .
```

then the following holds:

```
<someCurrency> rdf:type :Currency .
```

Other restrictions

- **`allValuesFrom`** could be replaced by:
  - **`someValuesFrom`**
    - e.g., I could have said: there should be a price given **in *at least one*** of those currencies
  - **`hasValue`**, when restricted to *one specific value*

Similar concept: cardinality restriction

- In a property restriction, the goal was to restrict the possible *values* of a property
- In a cardinality restriction, the *number* of relations with that property is restricted
  - "a book being on offer" could be characterized as having at *least one price property* (i.e., the price of the book has been established)

Cardinality restriction

```
:Book_on_sale rdfs:subClassOf [
      rdf:type                owl:Restriction;
      owl:onProperty          p:price;
      owl:minCardinality      "1"^^xsd:integer.
   ].
```

- could also be "**owl:cardinality**" or "**owl:maxCardinality**"

But: OWL is hard!

- The combination of class constructions with various restrictions is extremely powerful
- What we have so far follows the same logic as before
  - extend the basic RDF and RDFS possibilities with new features
  - define their semantics, ie, what they “mean” in terms of relationships
  - expect to infer new relationships based on those
- However… a full inference procedure is hard 
  - not implementable with simple rule engines, for example

Tradeoff between Expressive Power and Efficient Reasoning Support

- The richer the language is, the more inefficient the reasoning support becomes
- Sometimes it crosses the border of *noncomputability*
- We need a compromise:
  - A language supported by reasonably efficient reasoners
  - A language that can express large classes of ontologies and knowledge.

Reasoning About Knowledge in Ontology Languages

- Class membership
  - If x is an instance of a class C, and C is a subclass of D, then we can infer that x is an instance of D
- Equivalence of classes
  - If class A is equivalent to class B, and class B is equivalent to class C, then A is equivalent to C, too

Reasoning About Knowledge in Ontology Languages (2)

- Consistency
  - X instance of classes A and B, but A and B are **disjoint**
  - This is an indication of an error in the ontology
- Classification
  - Certain property-value pairs are a sufficient condition for membership in a class A; if an individual x satisfies such conditions, we can conclude that x must be an **instance of A**

Three Species of OWL

- W3C'sWeb Ontology Working Group defined OWL as three different sublanguages:
  - OWL Full
  - OWL DL
  - OWL Lite
- Each sublanguage geared toward fulfilling different aspects of requirements

OWL Full

- It uses all the OWL languages primitives
- It allows the combination of these primitives in arbitrary ways with RDF and RDF Schema
- OWL Full is fully upward-compatible with RDF, both syntactically and semantically
- OWL Full is so powerful that it is undecidable
  - No complete (or efficient) reasoning support

OWL DL

- OWL DL (Description Logic) is a sublanguage of OWL Full that restricts application of the constructors from OWL and RDF
  - Application of OWL's constructors' to each other is disallowed
  - Therefore it corresponds to a well studied description logic
- OWL DL permits efficient reasoning support
- **But** we lose full compatibility with RDF:
  - Not every RDF document is a legal OWL DL document.
  - Every legal OWL DL document is a legal RDF document.

OWL Lite

- An even further restriction limits OWL DL to a subset of the language constructors
  - E.g., OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality.
- The advantage of this is a language that is easier to
  - grasp, for users
  - implement, for tool builders
- The disadvantage is restricted expressivity

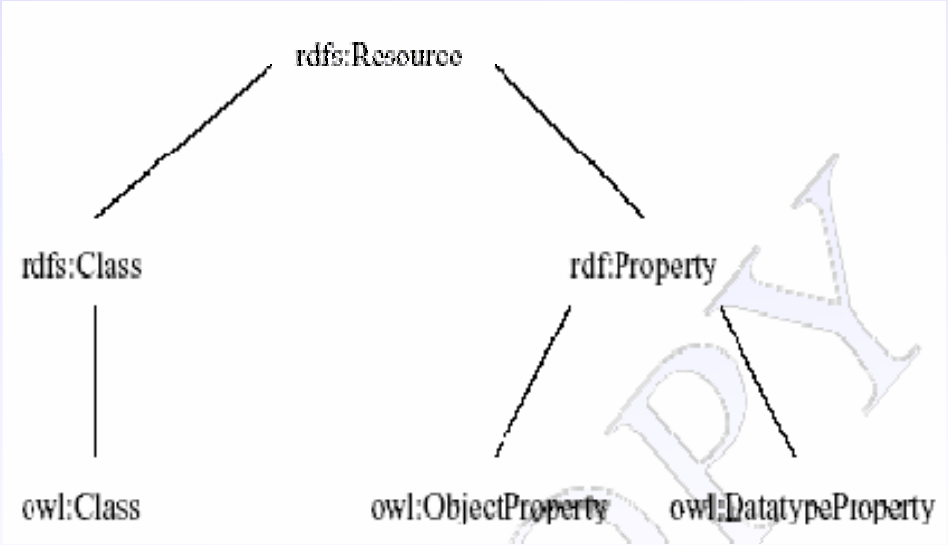
Upward Compatibility between OWL Species

- Every legal OWL Lite **ontology** is a legal OWL DL ontology
- Every legal OWL DL **ontology** is a legal OWL Full ontology

- Every valid OWL Lite **conclusion** is a valid OWL DL conclusion
- Every valid OWL DL **conclusion** is a valid OWL Full conclusion

OWL Compatibility with RDF Schema

- All varieties of OWL use RDF for their syntax
- Instances are declared as in RDF, using RDF descriptions
- and typing information OWL constructors are specialisations of their RDF counterparts



OWL Compatibility with RDF Schema (2)

- Semantic Web design aims at downward compatibility with corresponding reuse of software across the various layers
- The advantage of full downward compatibility for OWL is only achieved for OWL Full, at the cost of computational intractability

Ontology development

- The hard work is to *create* the ontologies
  - requires a good knowledge of the area to be described
  - some communities have good expertise already (e.g., librarians)
  - *OWL is just a tool to formalize ontologies*
  - large scale ontologies are often developed in a community process
- Ontologies should be *shared* and *reused*
  - can be via the simple namespace mechanisms…
  - …or via explicit imports
- Applications can also be developed with very small ontologies, though

Ontologies examples

- eClassOwl: eBusiness ontology for products and services, 75,000 classes and 5,500 properties
- National Cancer Institute's ontology: about 58,000 classes
- Open Biomedical Ontologies Foundry: a collection of ontologies, including the Gene Ontology to describe gene and gene product attributes in any organism or protein sequence and annotation terminology and data (UniProt)
- BioPAX: for biological pathway data

Editing OWL Ontologies with Protégé

This Tutorial

- Introduction to OWL, the Semantic Web, and the Protégé OWL Plugin
- Theory + Walkthrough

Overview

The Semantic Web and OWL

Basic OWL

Interactive: Classes, Properties

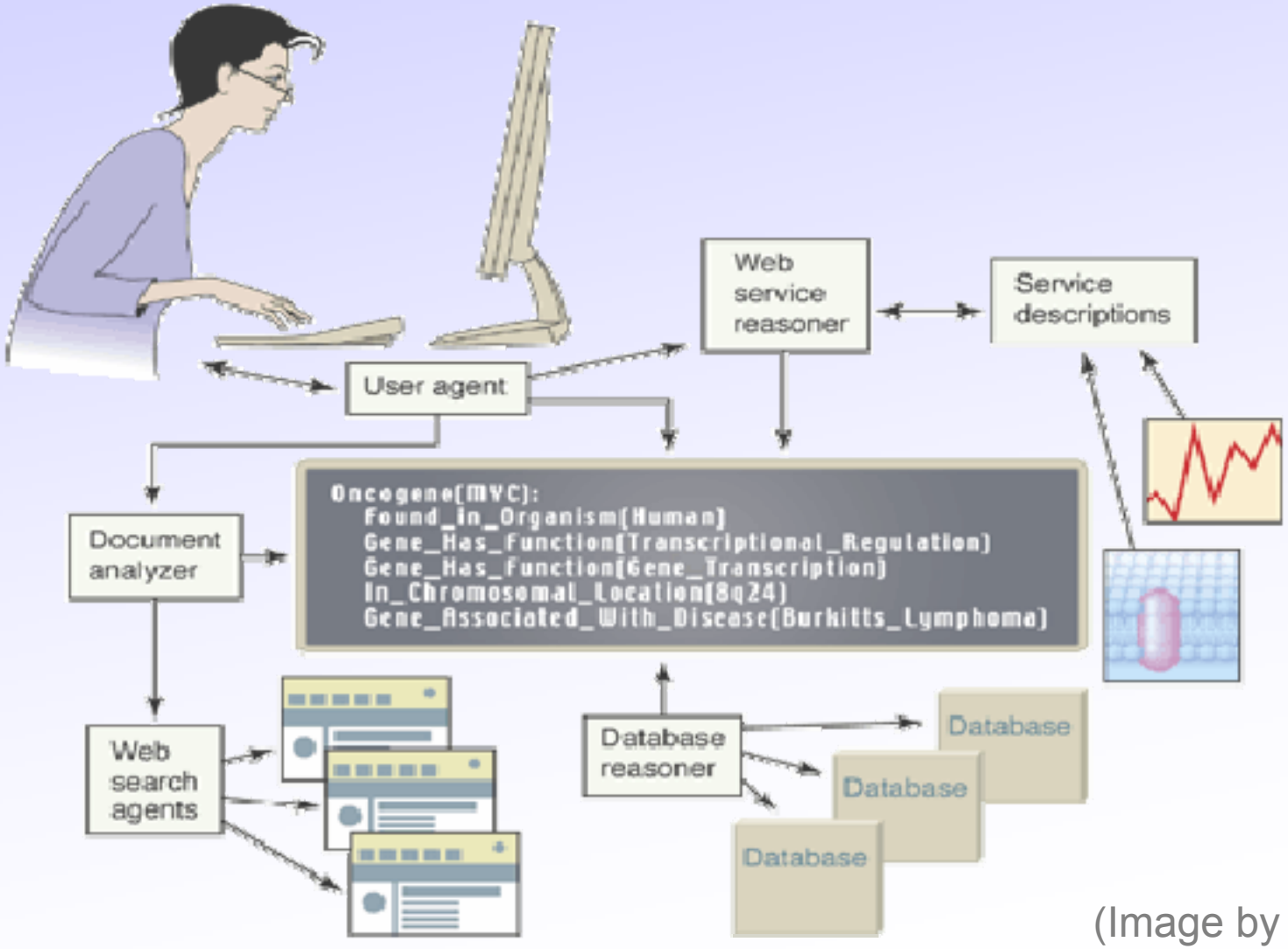
Advanced OWL

Interactive: Class Descriptions

Creating Semantic Web Contents

The Semantic Web

Shared ontologies help to exchange data and meaning between web-based services

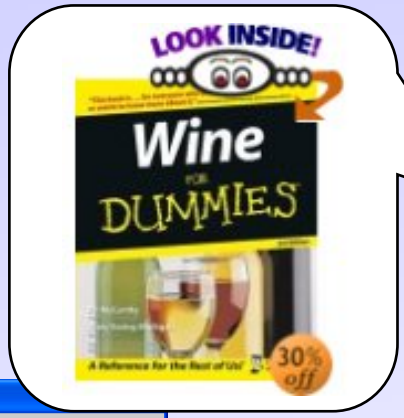


(Image by Jim Hendler)

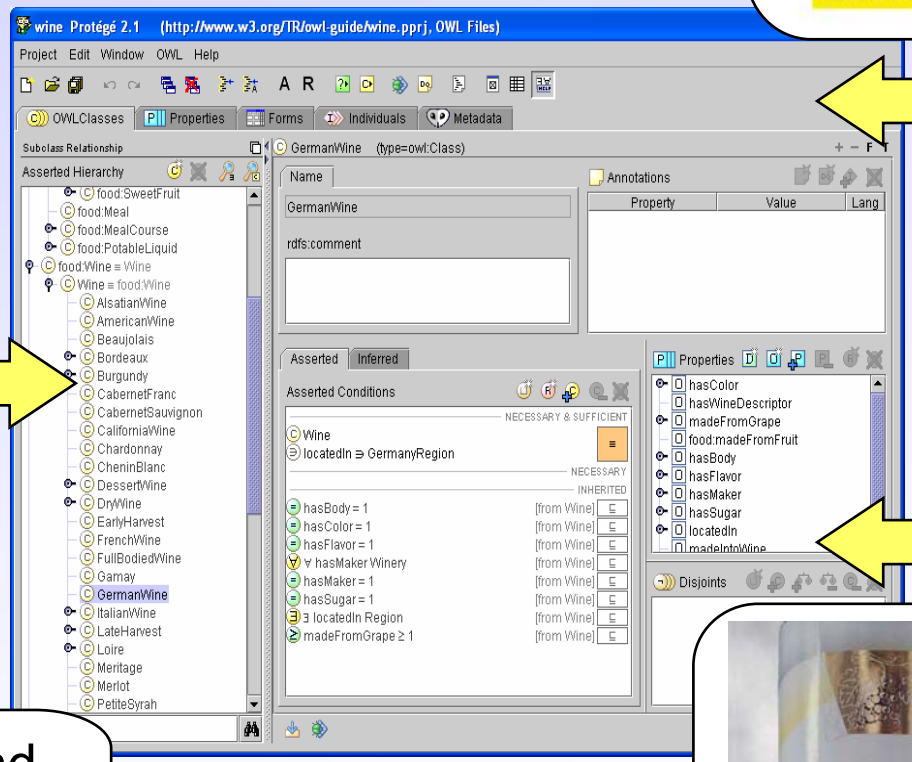
Wine Example Scenario

Tell me what wines I should buy to serve with each course of the following menu.

Wine Agent



Books Agent



I recommend Chardonney or DryRiesling

Grocery Agent

Ontologies in the Semantic Web

- Provide shared data structures to exchange information between agents
- Can be explicitly used as annotations in web sites
- Can be used for knowledge-based services using other web resources
- Can help to structure knowledge to build domain models (for other purposes)

OWL

- Web Ontology Language
- Official W3C Standard since Feb 2004
- Based on predecessors (DAML+OIL)

- A Web Language: Based on RDF(S)
- An Ontology Language: Based on logic

OWL Ontologies

- What's inside an OWL ontology
  - Classes + class-hierarchy
  - Properties (Slots) / values
  - Relations between classes (inheritance, disjoints, equivalents)
  - Restrictions on properties (type, cardinality)
  - Characteristics of properties (transitive, …)
  - Annotations
  - Individuals
- Reasoning tasks: classification, consistency checking

OWL Use Cases

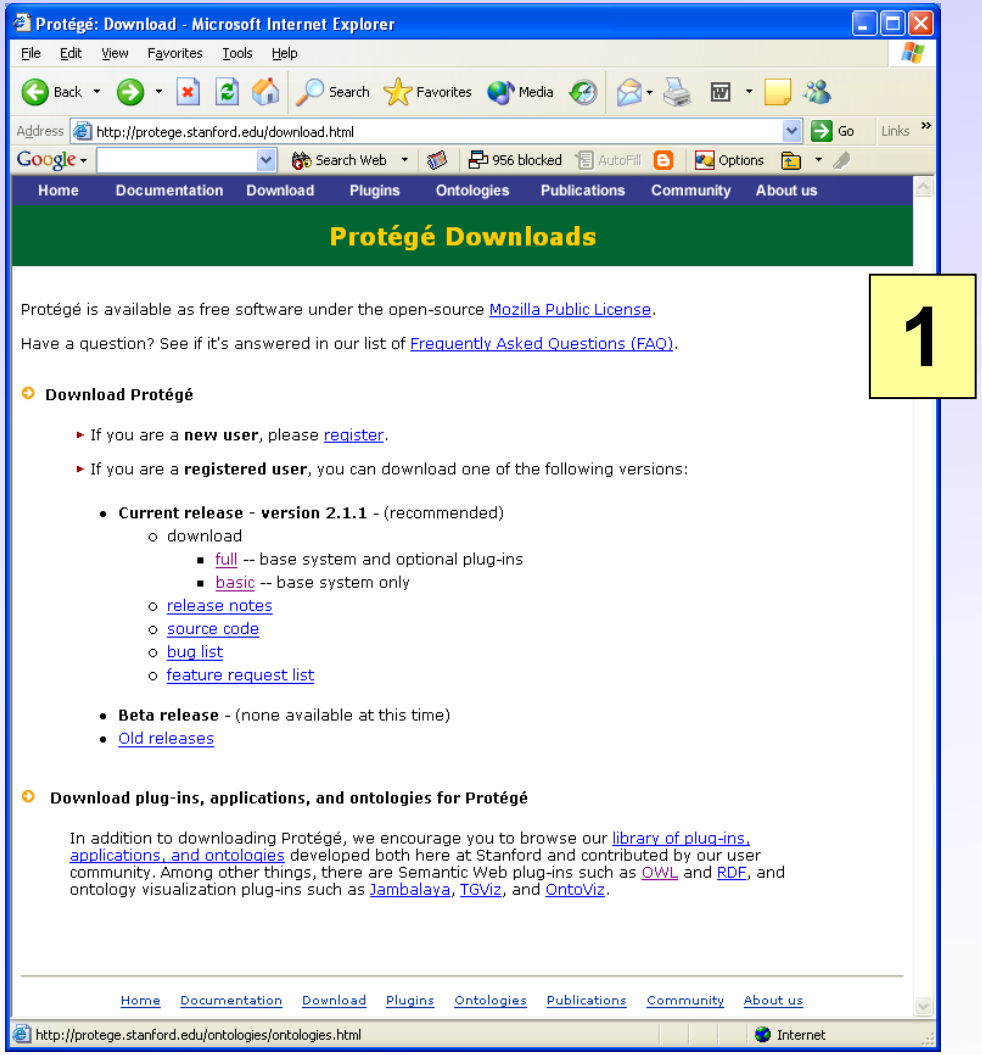
- At least two different user groups
  - OWL used as data exchange language (define interfaces of services and agents)
  - OWL used for terminologies or knowledge models
- OWL DL is the subset of OWL (Full) that is optimized for reasoning and knowledge modeling

Protégé OWL Plugin

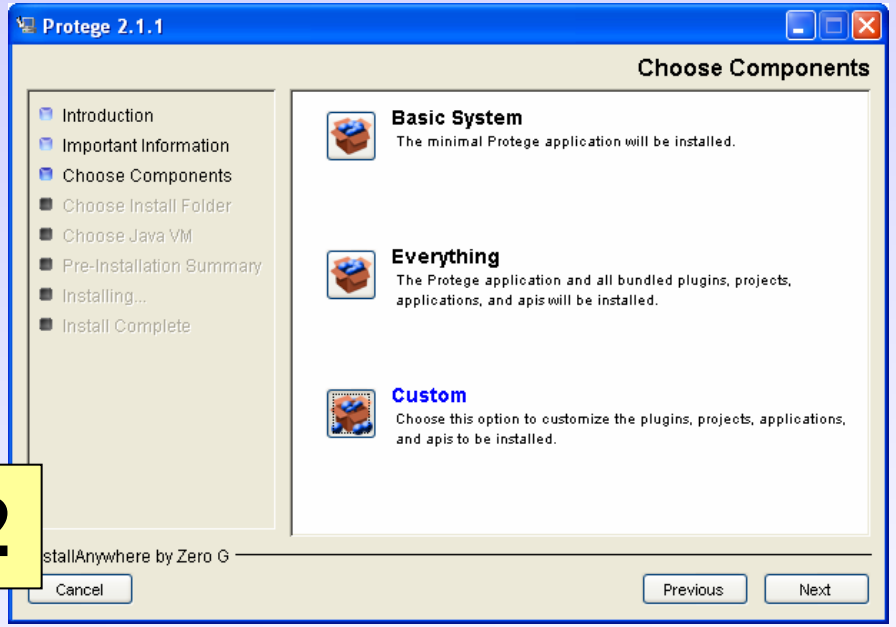
- Extension of Protégé for handling OWL ontologies
- Project started in April 2003
- Features
  - Loading and saving OWL files & databases
  - Graphical editors for class expressions
  - Access to description logics reasoners
  - Powerful platform for hooking in custom-tailored components

Installation

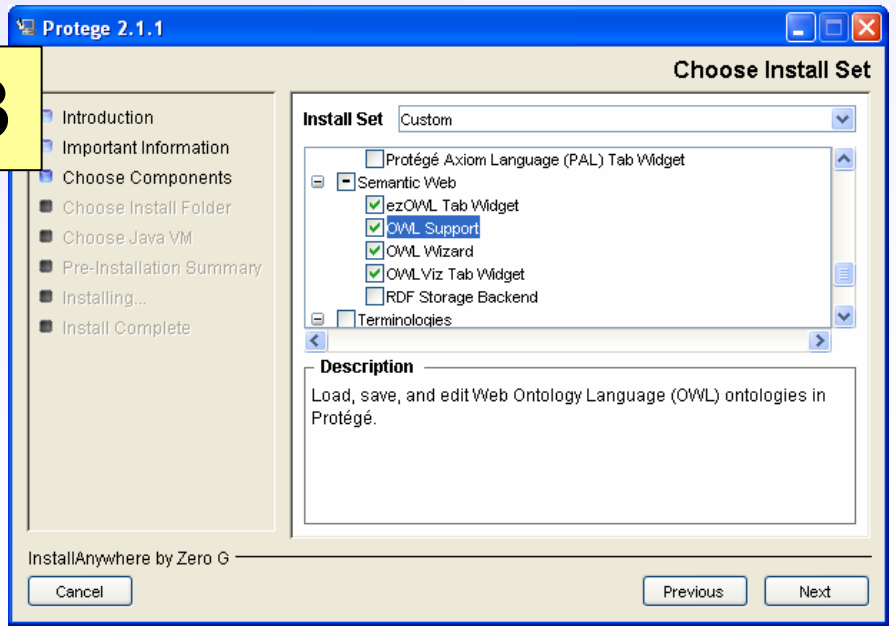
Bundled in Protégé 2.1 (Full):



1



2



3

Tutorial Scenario

- Semantic Web for Tourism/Traveling
- Goal: Find matching holiday destinations for a customer

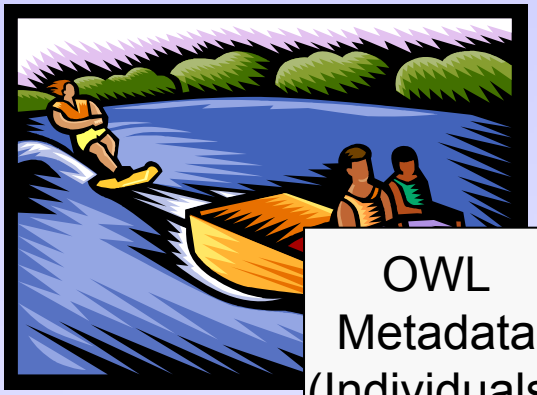
Scenario Architecture

- A search problem: Match customer's expectations with potential destinations
- Required: Web Service that exploits formal information about the available destinations
  - Accomodation (Hotels, B&B, Camping, ...)
  - Activities (Sightseeing, Sports, ...)

Tourism Semantic Web

- Open World:
  - New hotels are being added
  - New activities are offered
- Providers publish their services dynamically
- Standard format / grounding is needed
  → Tourism Ontology

Tourism Semantic Web



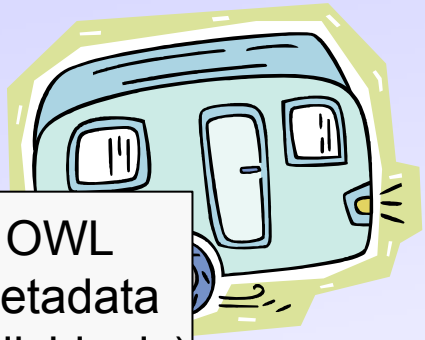
OWL Metadata (Individuals)

Tourism Ontology

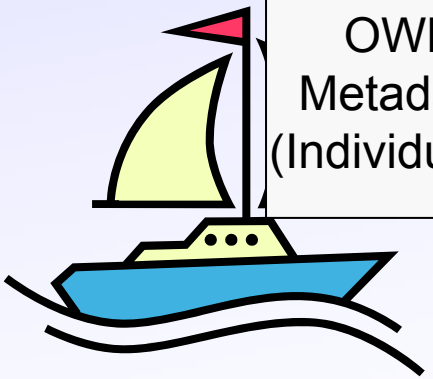
Destination

Activity

Accomodation



OWL Metadata (Individuals)



OWL Metadata (Individuals)

Web Services

OWL Metadata (Individuals)

OWL (in Protégé)

- Individuals (e.g., “FourSeasons”)

- Properties
  - ObjectProperties (references)
  - DatatypeProperties (simple values)

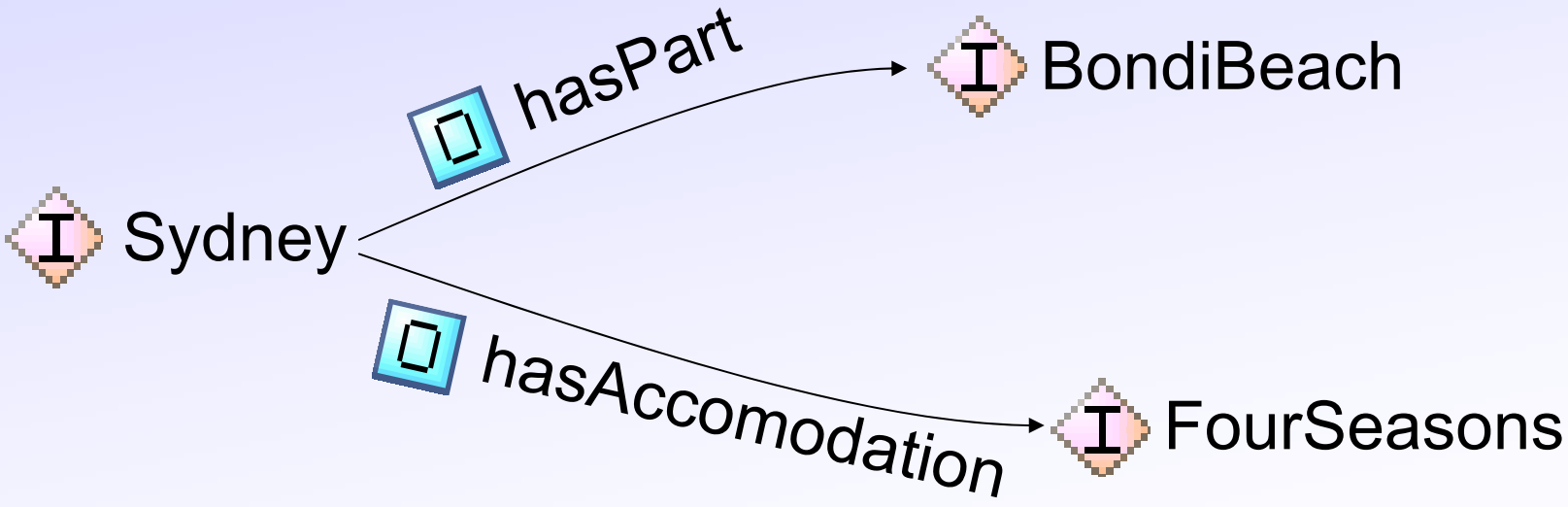
- Classes (e.g., “Hotel”)

Individuals

- Represent objects in the domain
- Specific things
- Two names could represent the same “real-world” individual

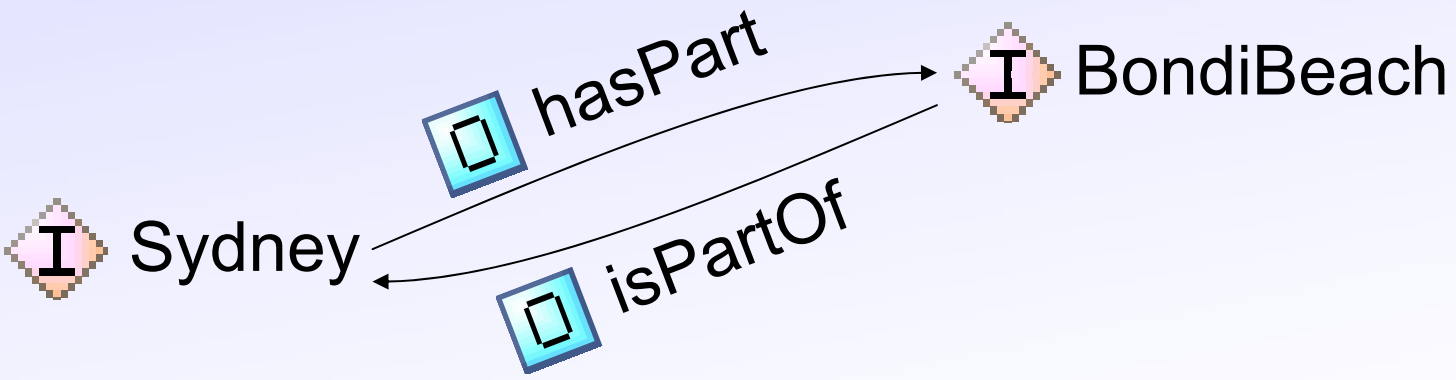
ObjectProperties

- Link two individuals together
- Relationships (0..n, n..m)



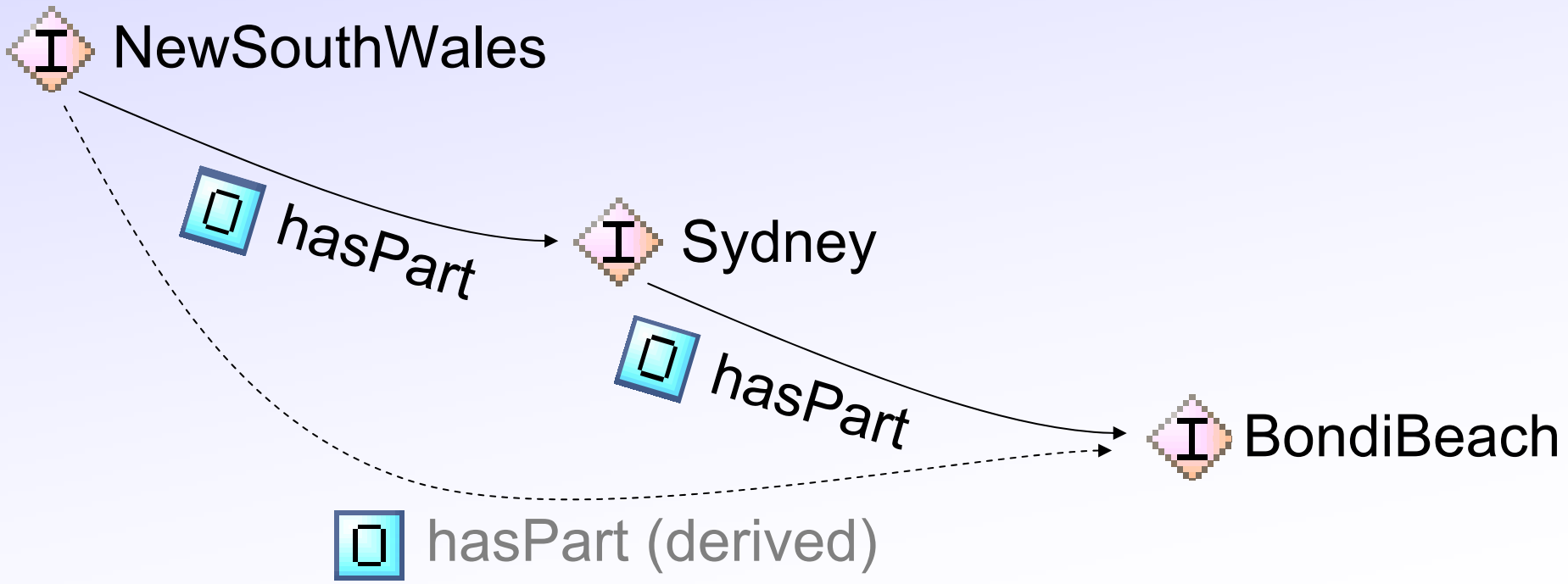
Inverse Properties

- Represent bidirectional relationships
- Adding a value to one property also adds a value to the inverse property



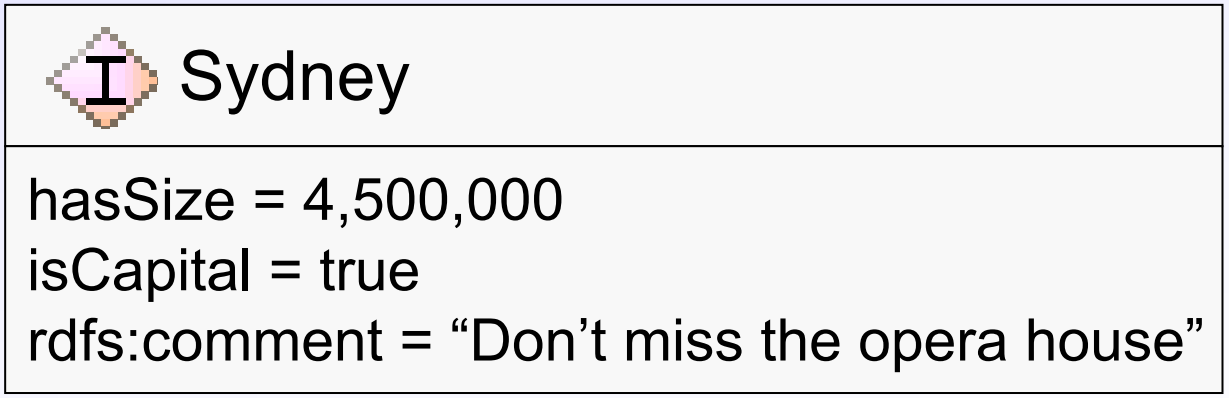
Transitive Properties

- If A is related to B and B is related to C then A is also related to C
- Often used for part-of relationships



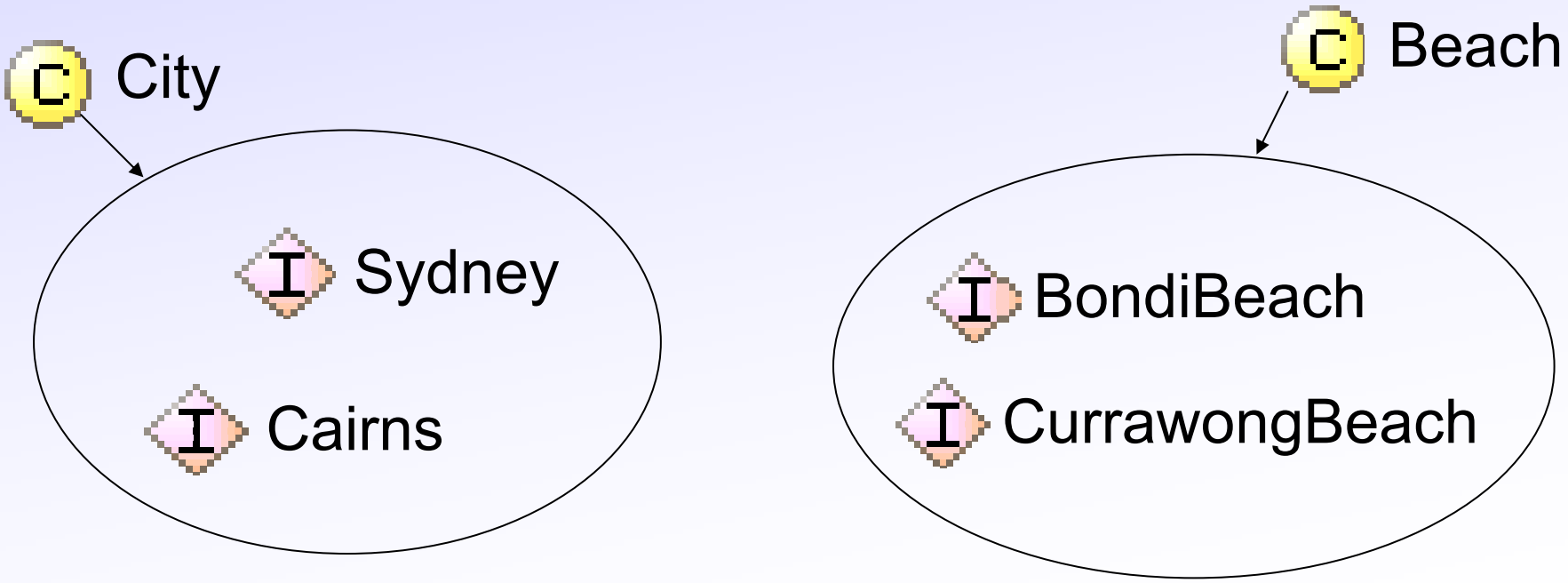
DatatypeProperties

- Link individuals to primitive values (integers, floats, strings, booleans etc)
- Often: AnnotationProperties without formal "meaning"



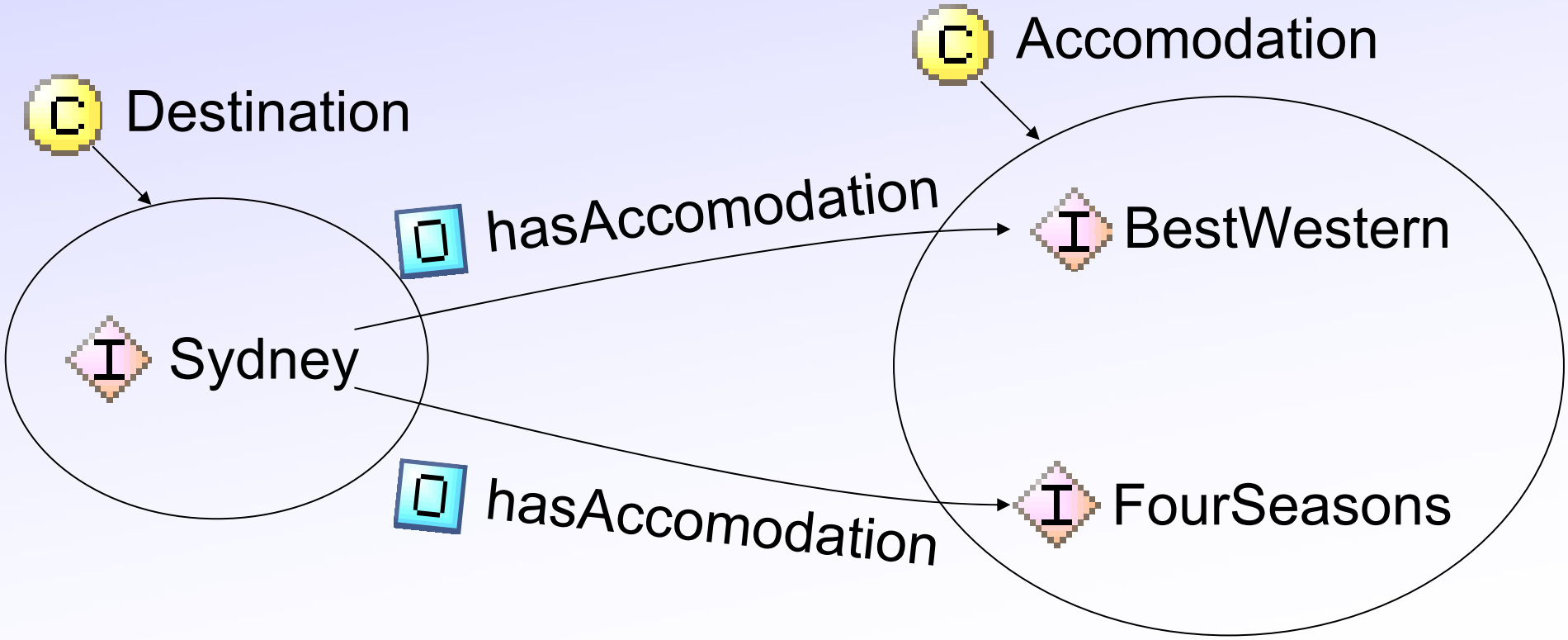
Classes

- Sets of individuals with common characteristics
- Individuals are *instances* of at least one class



Range and Domain

- Property characteristics
  - Domain: “left side of relation” (Destination)
  - Range: “right side” (Accomodation)

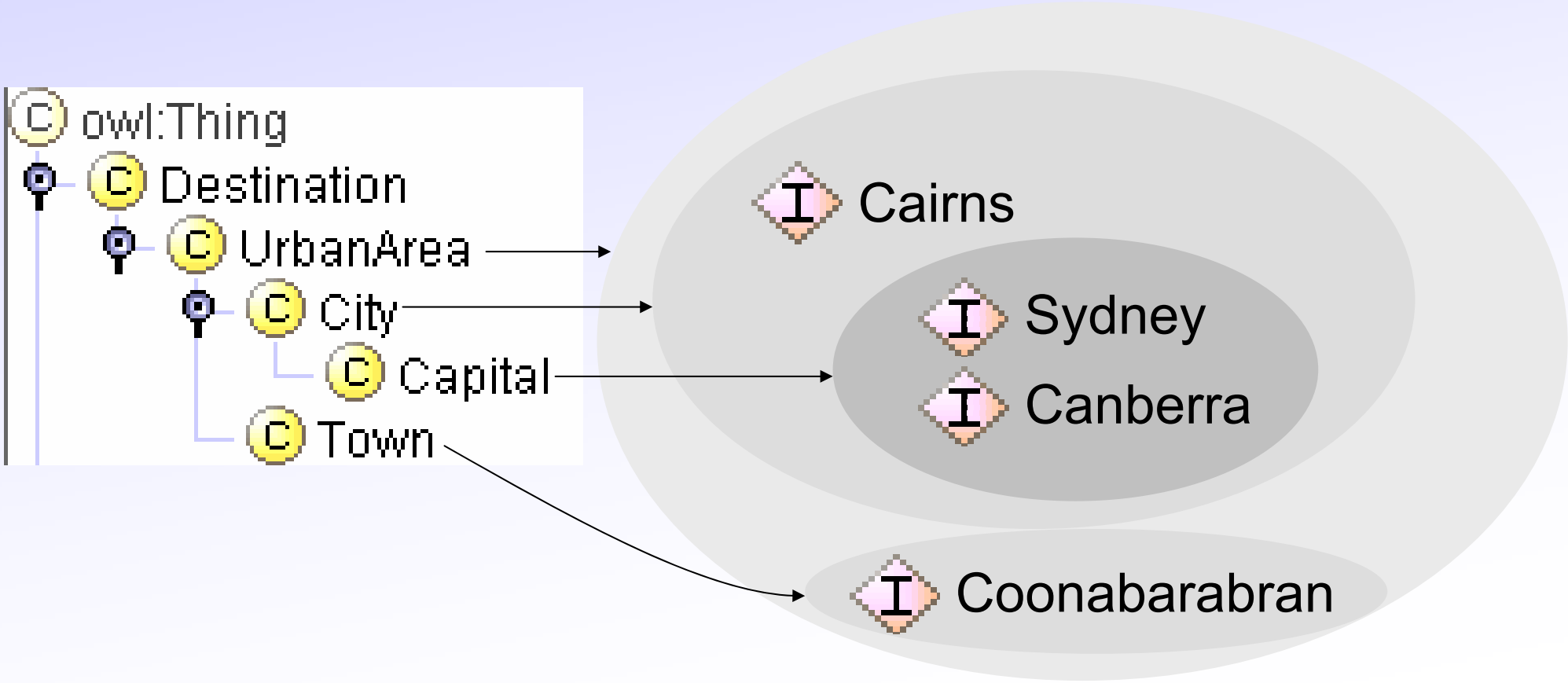


Domains

- Individuals can only take values of properties that have matching domain
  - “Only Destinations can have Accomodations”
- Domain can contain multiple classes
- Domain can be undefined:
  Property can be used everywhere

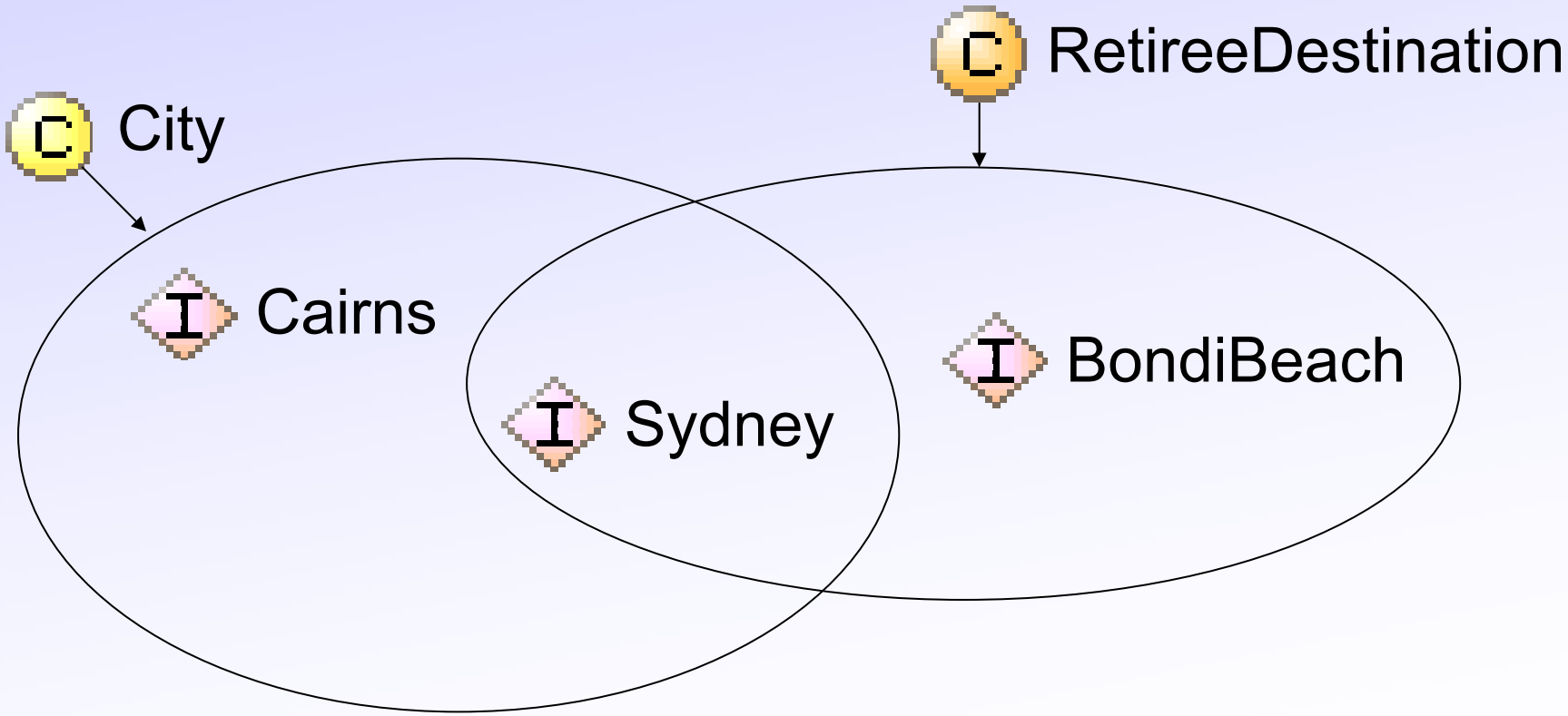
Superclass Relationships

- Classes can be organized in a hierarchy
- Direct instances of subclass are also (indirect) instances of superclasses



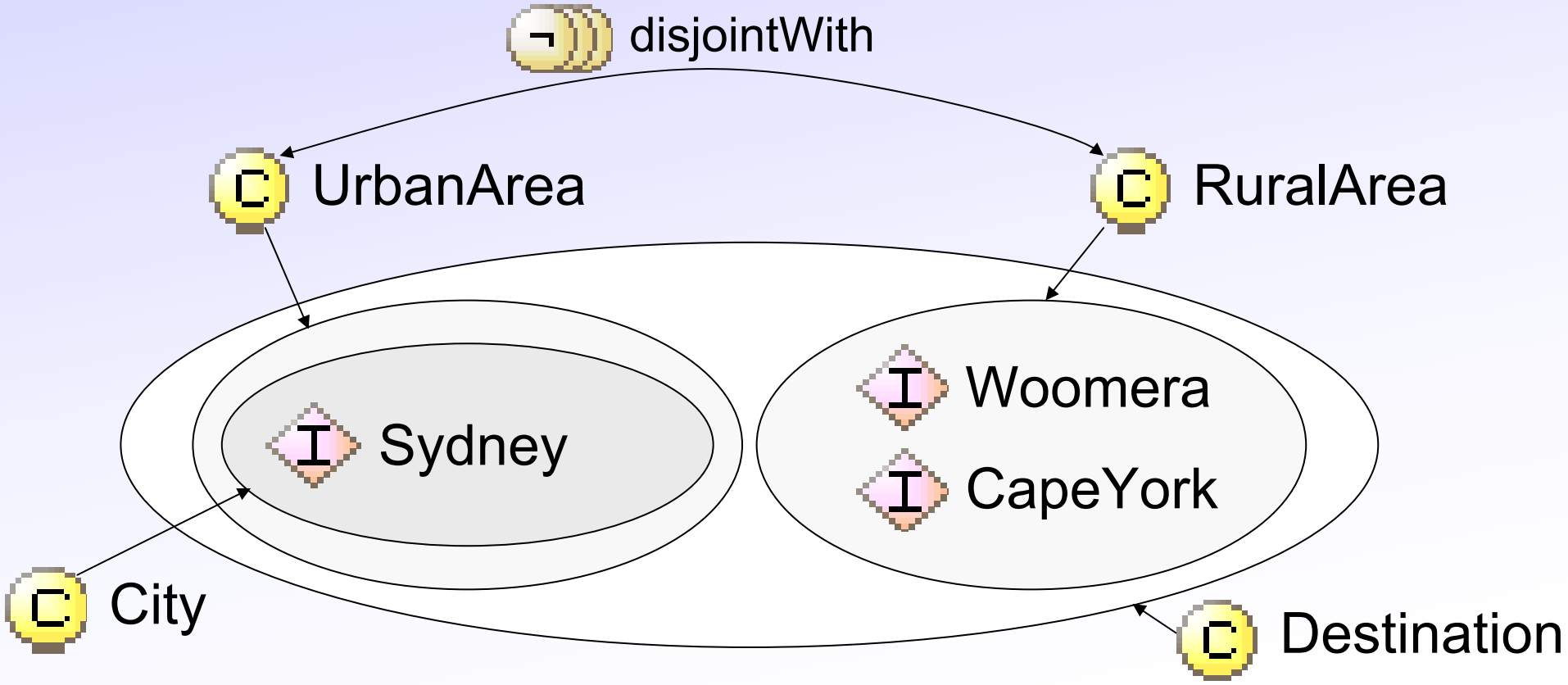
Class Relationships

- Classes can overlap arbitrarily

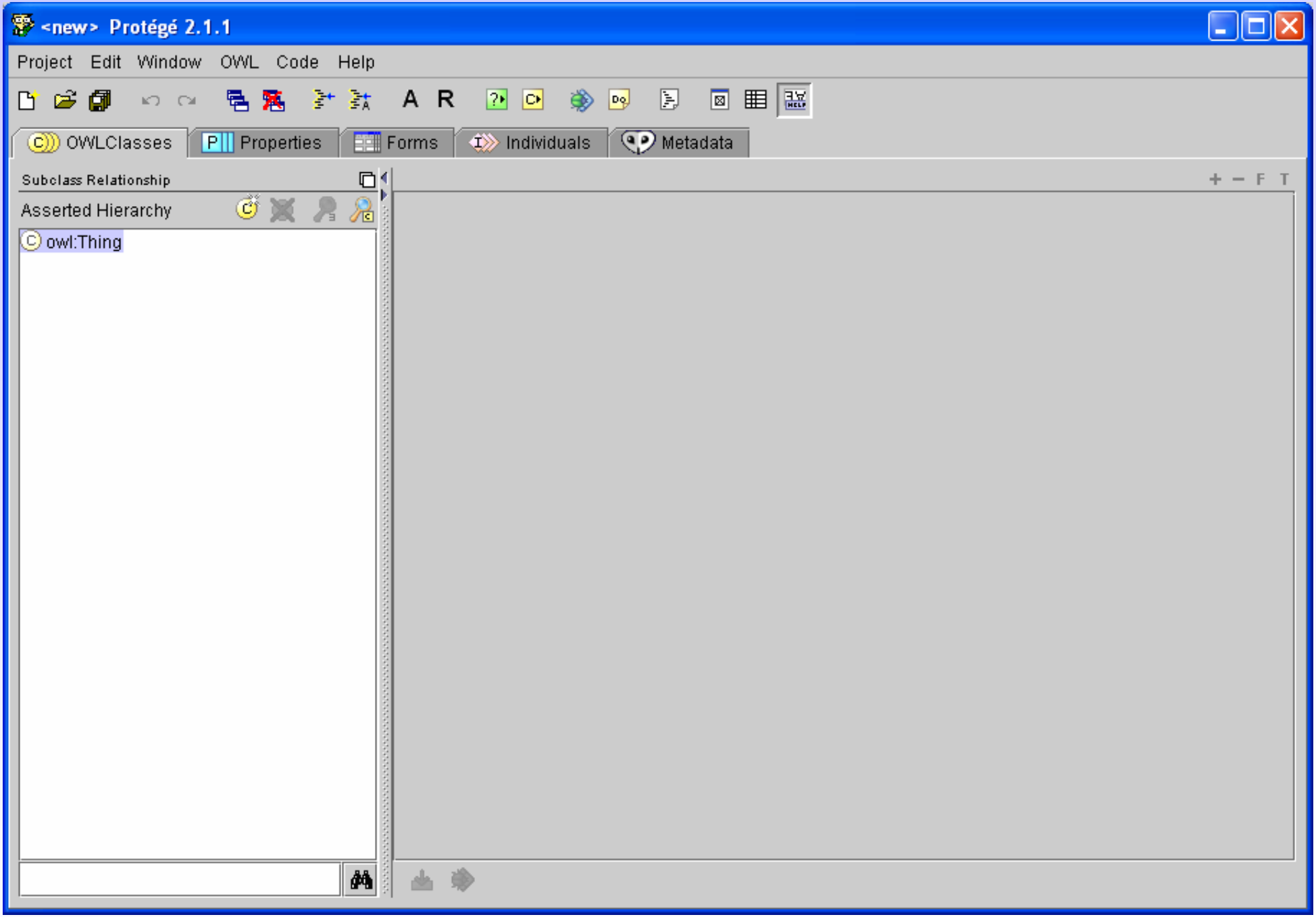


Class Disjointness

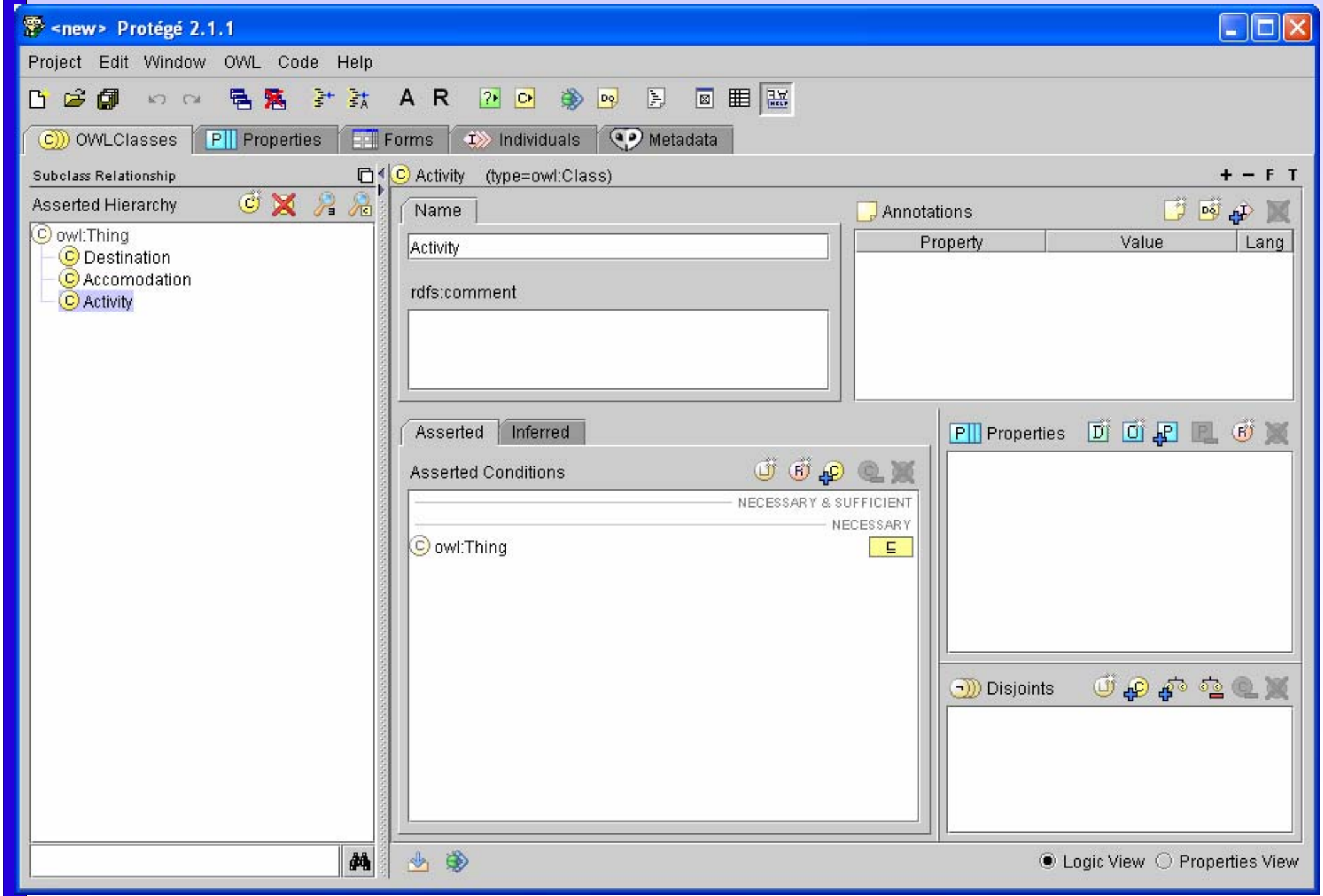
- All classes could potentially overlap
- In many cases we want to make sure they don't share instances



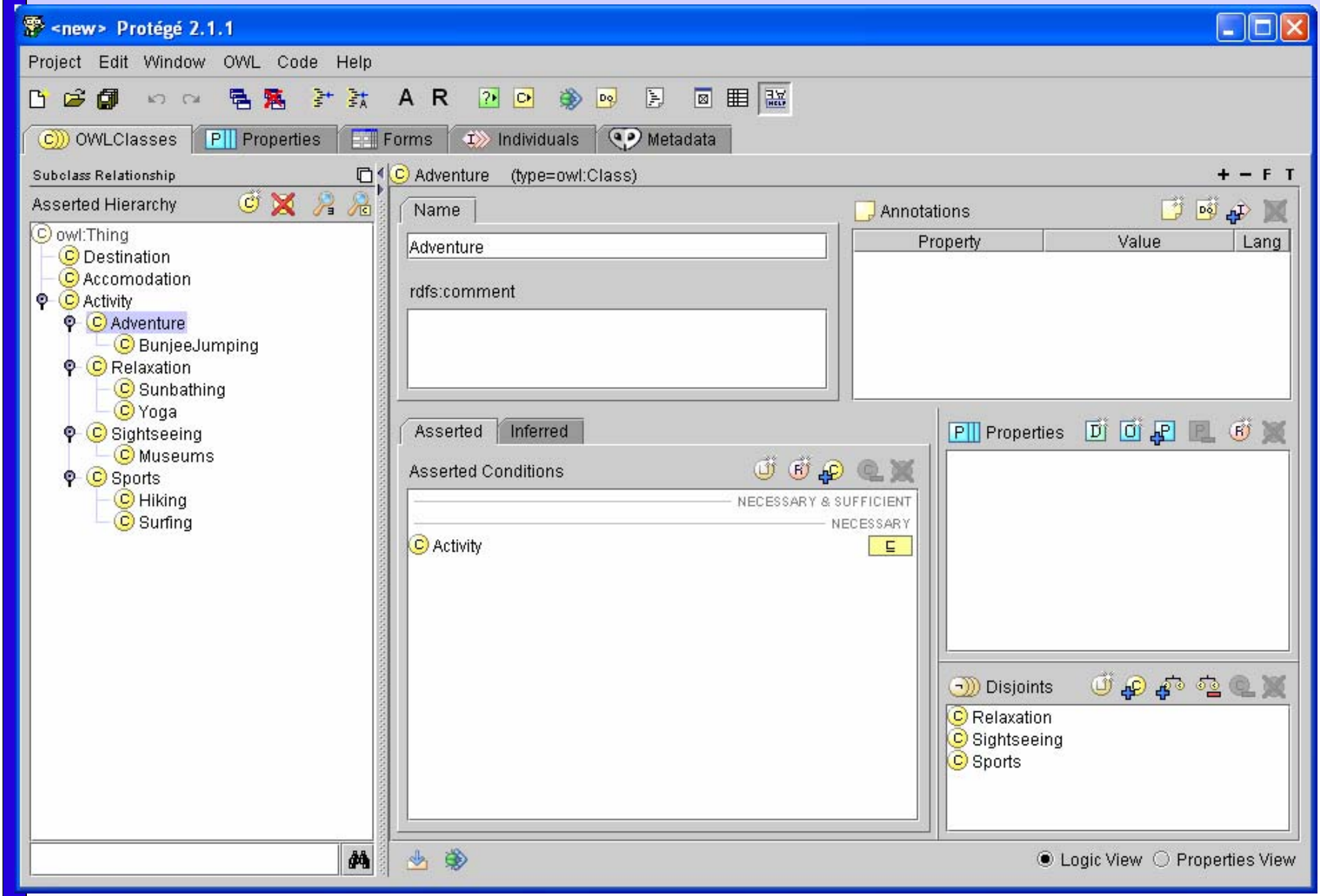
(Create a new OWL project)



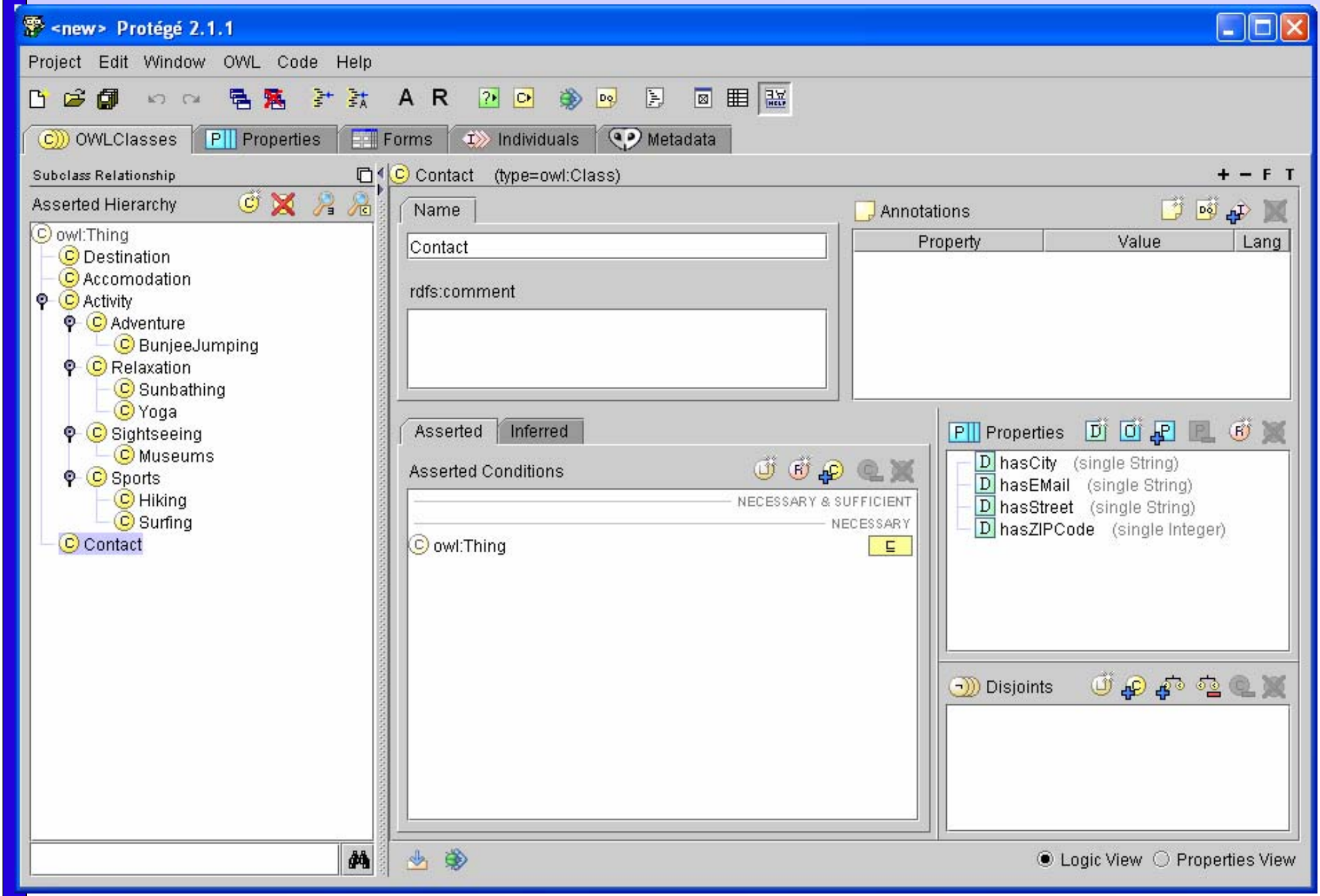
(Create simple classes)



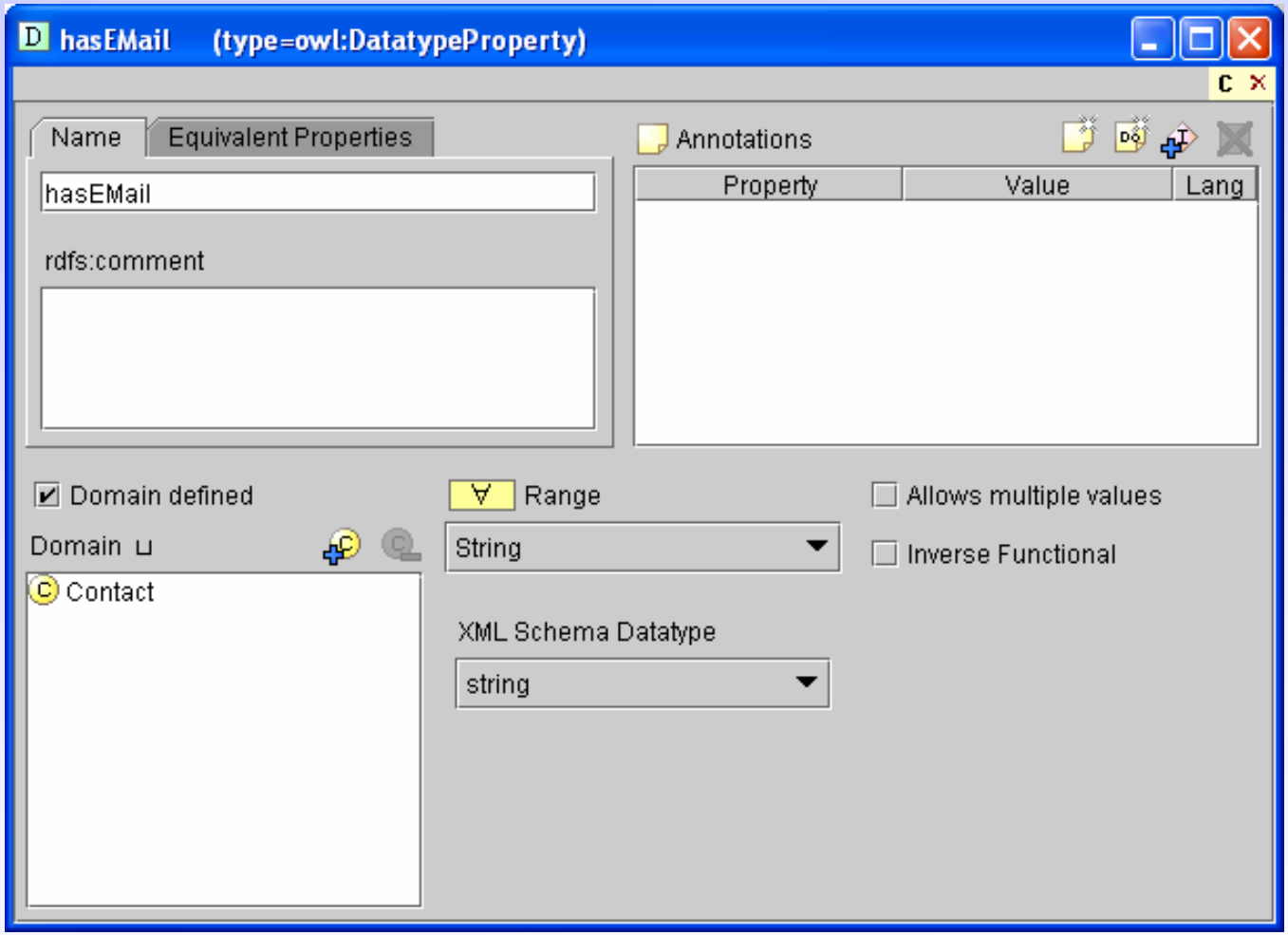
(Create class hierarchy and set disjoints)



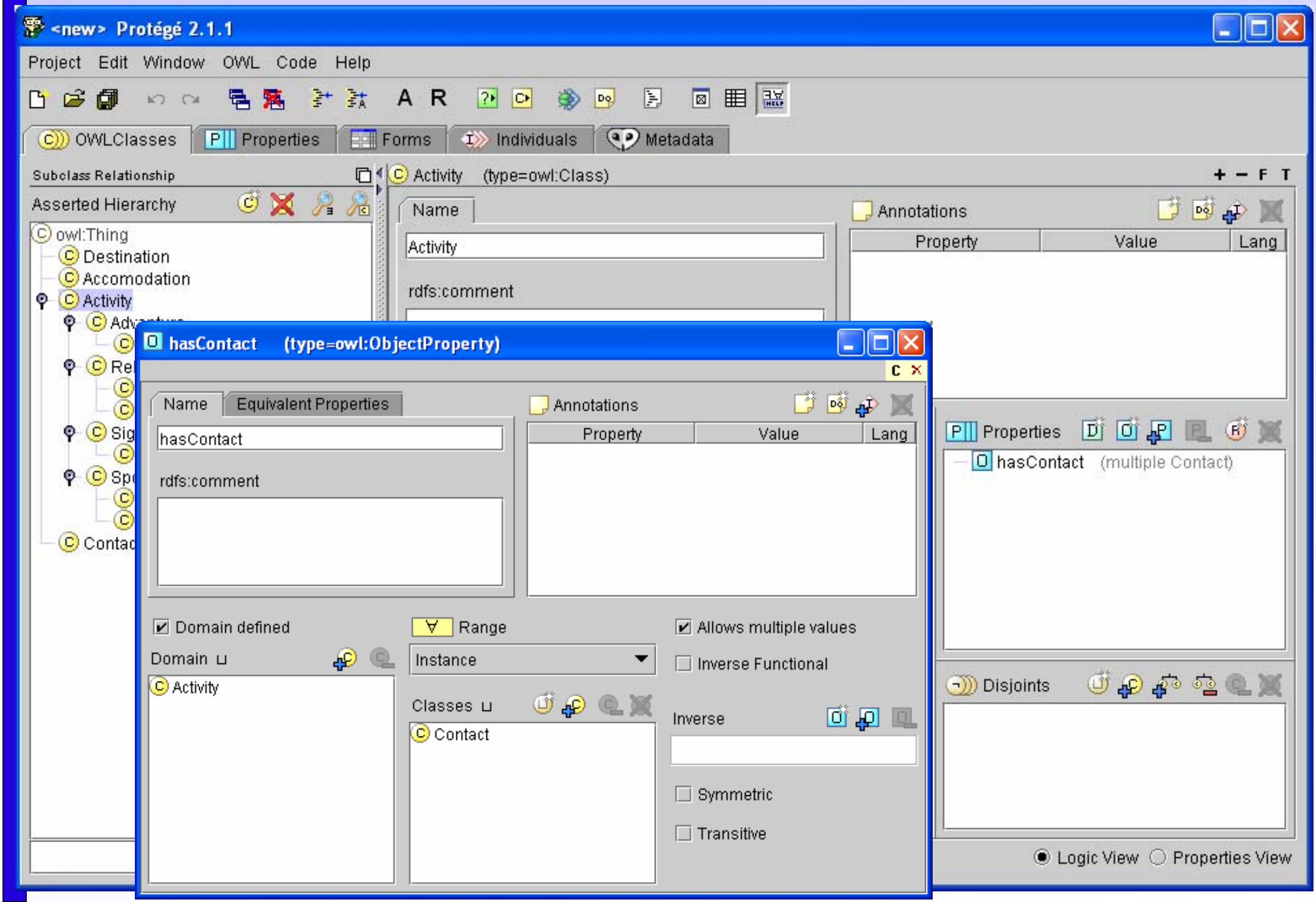
(Create Contact class with datatype properties)



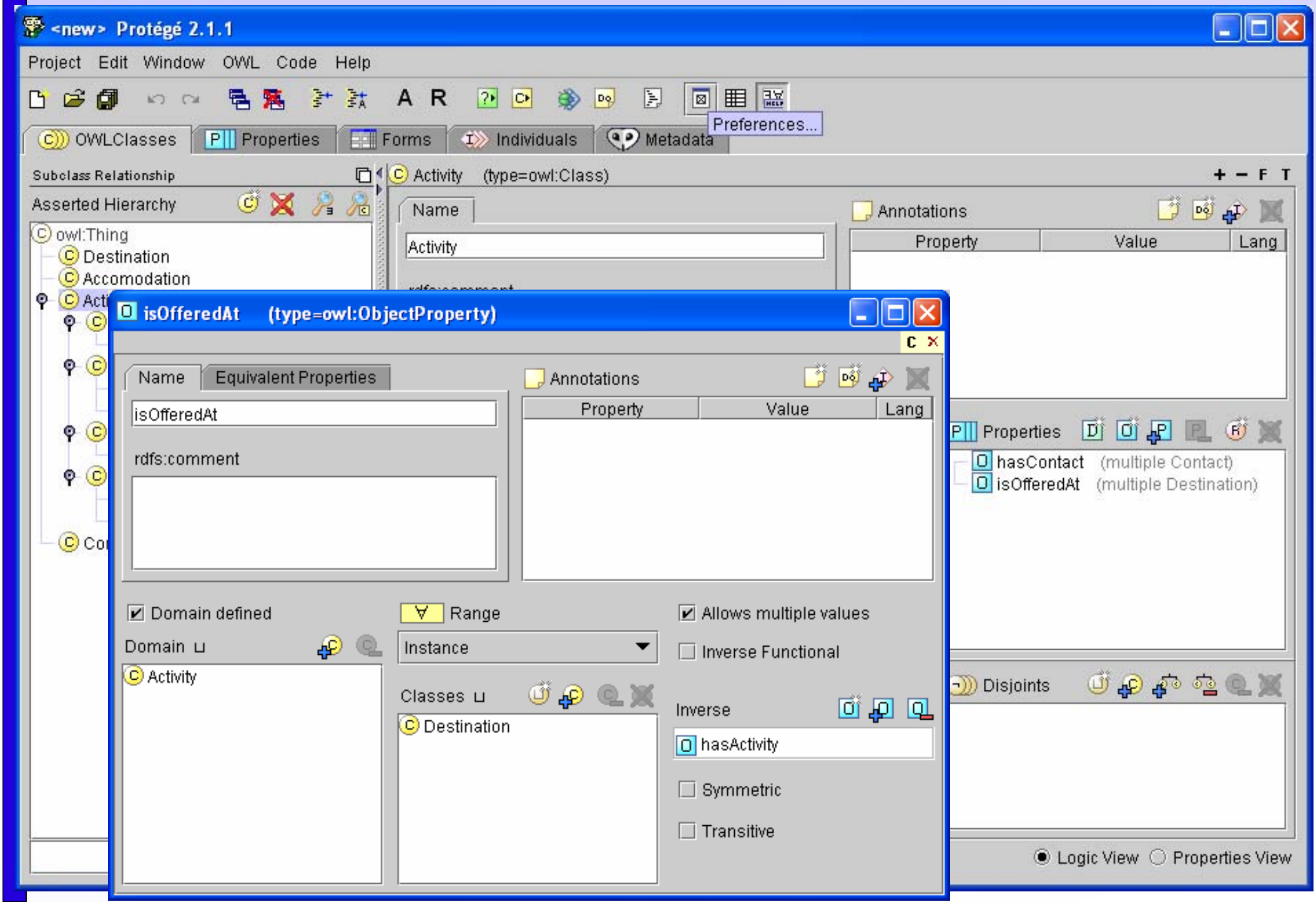
(Edit details of datatype properties)



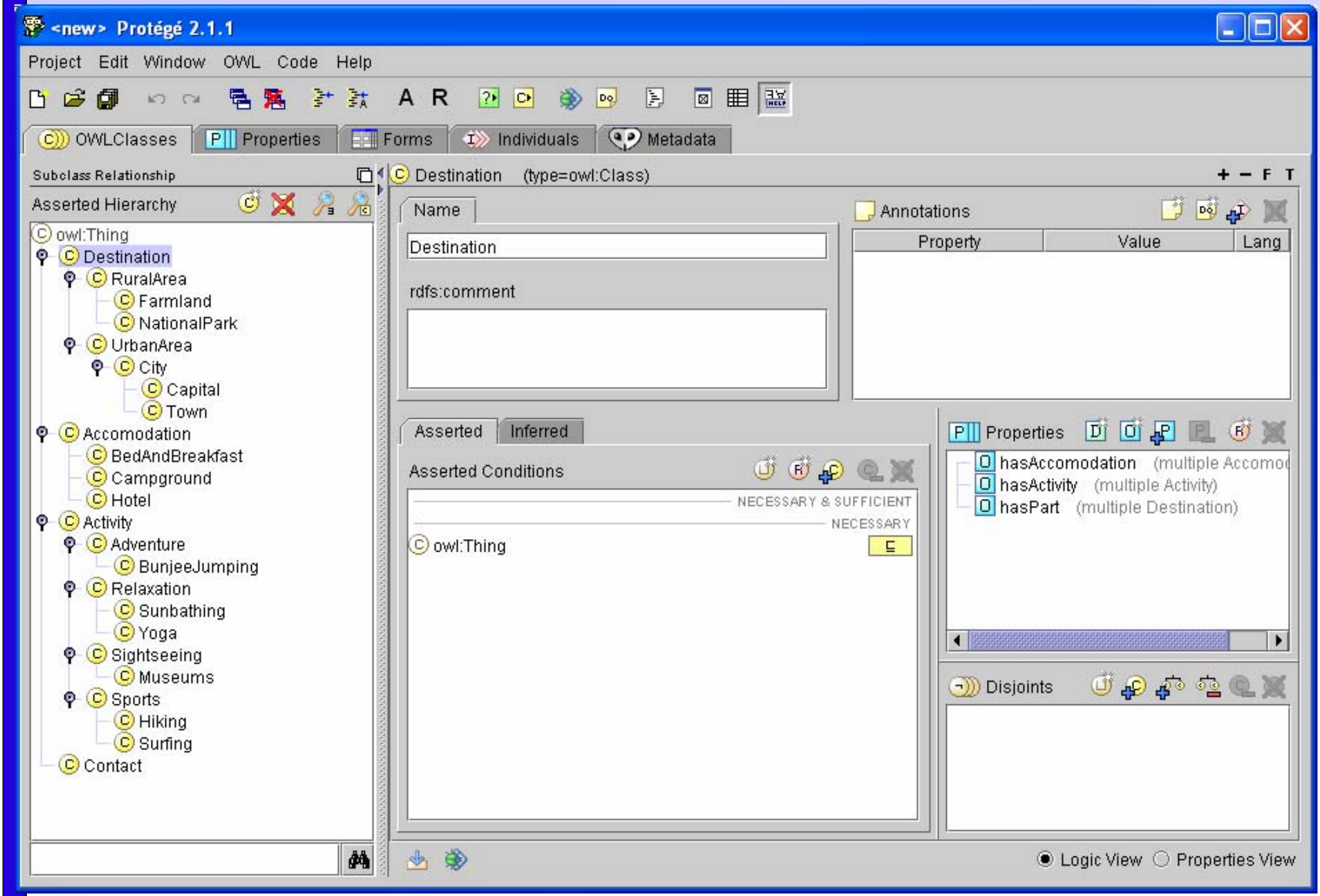
(Create an object property hasContact)



(Create an object property with inverse)

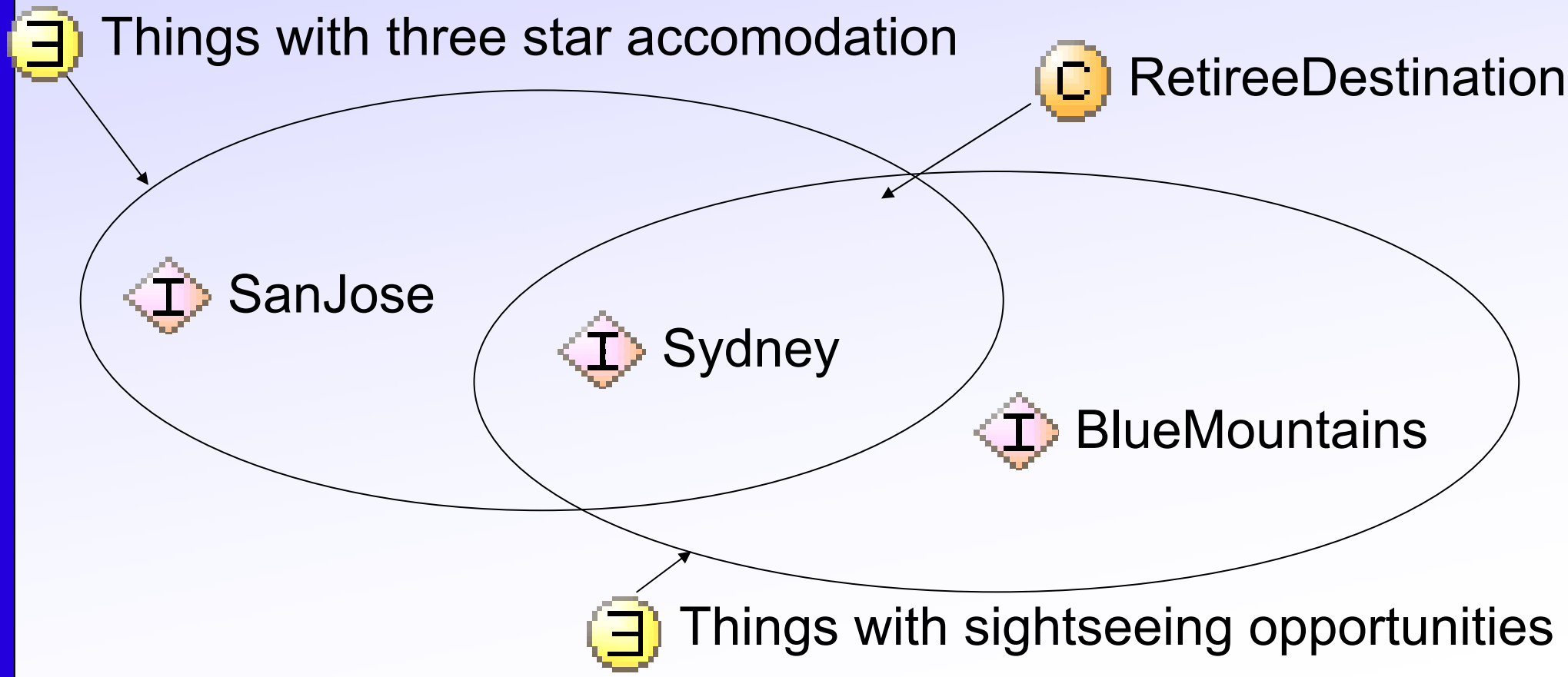


(Create the remaining classes and properties)



Class Descriptions

- Classes can be described by their logical characteristics
- Descriptions are “anonymous classes”



Class Descriptions

- Define the “meaning” of classes
- Anonymous class expressions are used
  - “All national parks have campgrounds.”
  - “A backpackers destination is a destination that has budget accomodation and offers sports or adventure activities.”
- Expressions mostly restrict property values (OWL Restrictions)

Class Descriptions: Why?

- Based on OWL's Description Logic support
- Formalize intentions and modeling decisions (comparable to test cases)
- Make sure that individuals fulfill conditions
- Tool-supported reasoning

Reasoning with Classes

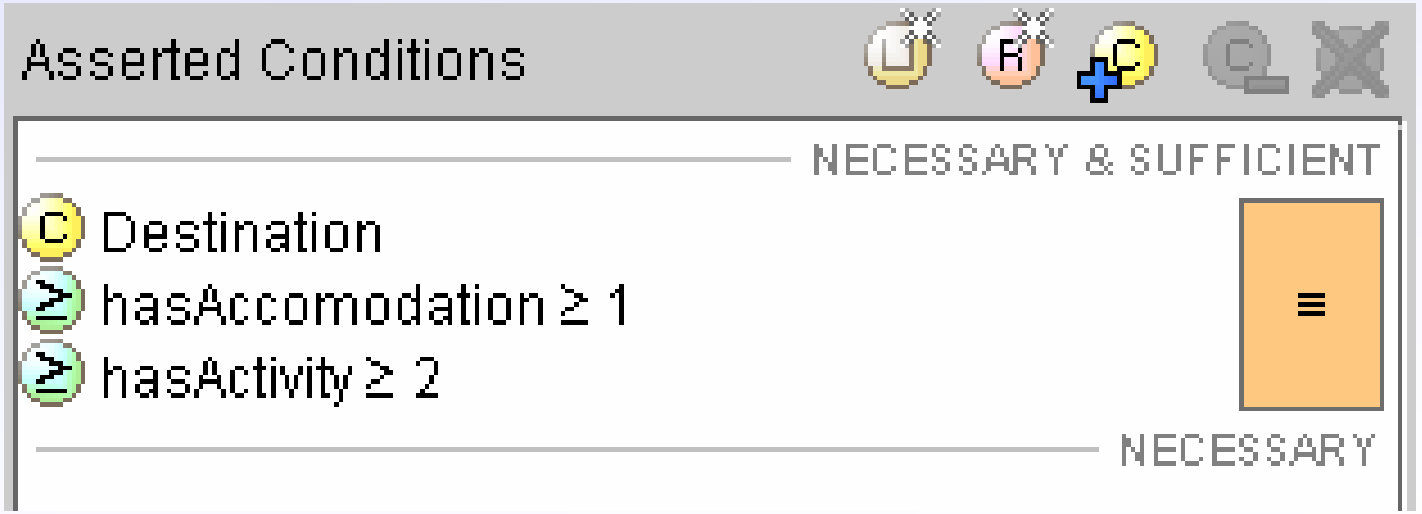
- Tool support for three types of reasoning exists:
  - Consistency checking:
    Can a class have any instances?
  - Classification:
    Is A a subclass of B?
  - Instance classification:
    Which classes does an individual belong to?
- For Protégé we recommend Pellet
  (but other tools with DIG support work too)
- Run pellet.bat dig and keep reasoner window open

Restrictions (Overview)

- Define a condition for property values
  -  allValuesFrom
  -  someValuesFrom
  -  hasValue
  -  minCardinality
  -  maxCardinality
  -  cardinality
- An anonymous class consisting of all individuals that fulfill the condition

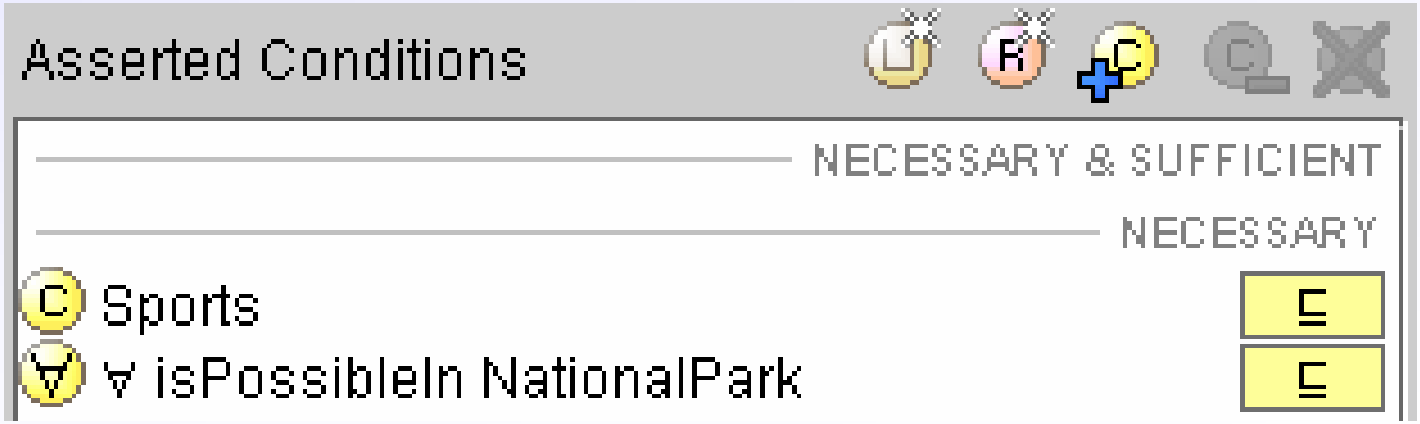
Cardinality Restrictions

- Meaning: The property must have at least/at most/exactly x values
-  is the shortcut for  and 
- Example: A FamilyDestination is a Destination that has at least one Accomodation and at least 2 Activities



allValuesFrom Restrictions

- Meaning: All values of the property must be of a certain type
- Warning: Also individuals with no values fulfill this condition (trivial satisfaction)
- Example: Hiking is a Sport that is only possible in NationalParks

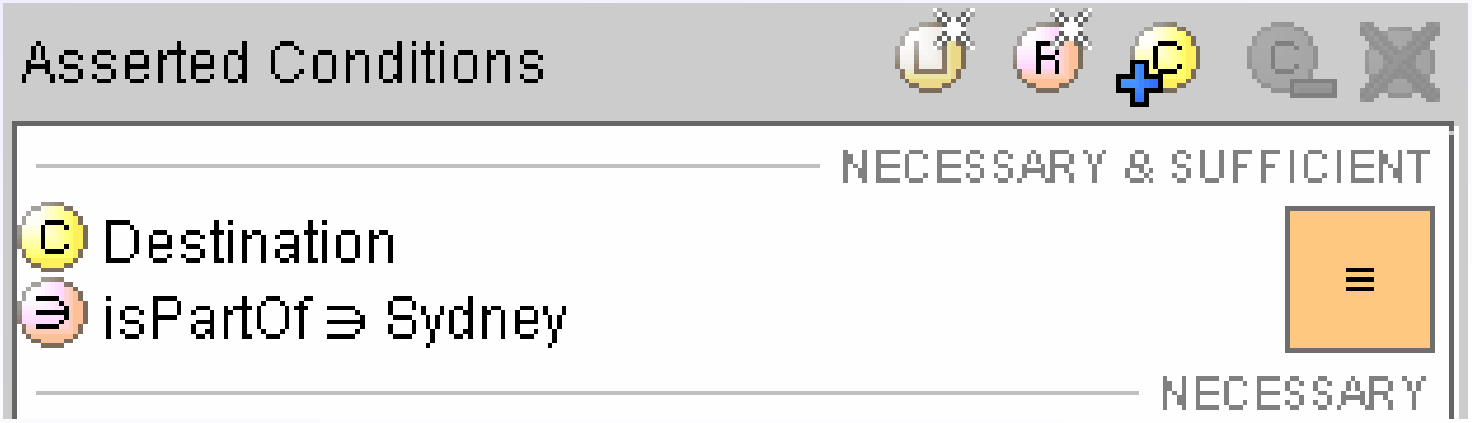


someValuesFrom Restrictions

- Meaning: At least one value of the property must be of a certain type
- Others may exist as well
- Example: A NationalPark is a RuralArea that has at least one Campground and offers at least one Hiking opportunity

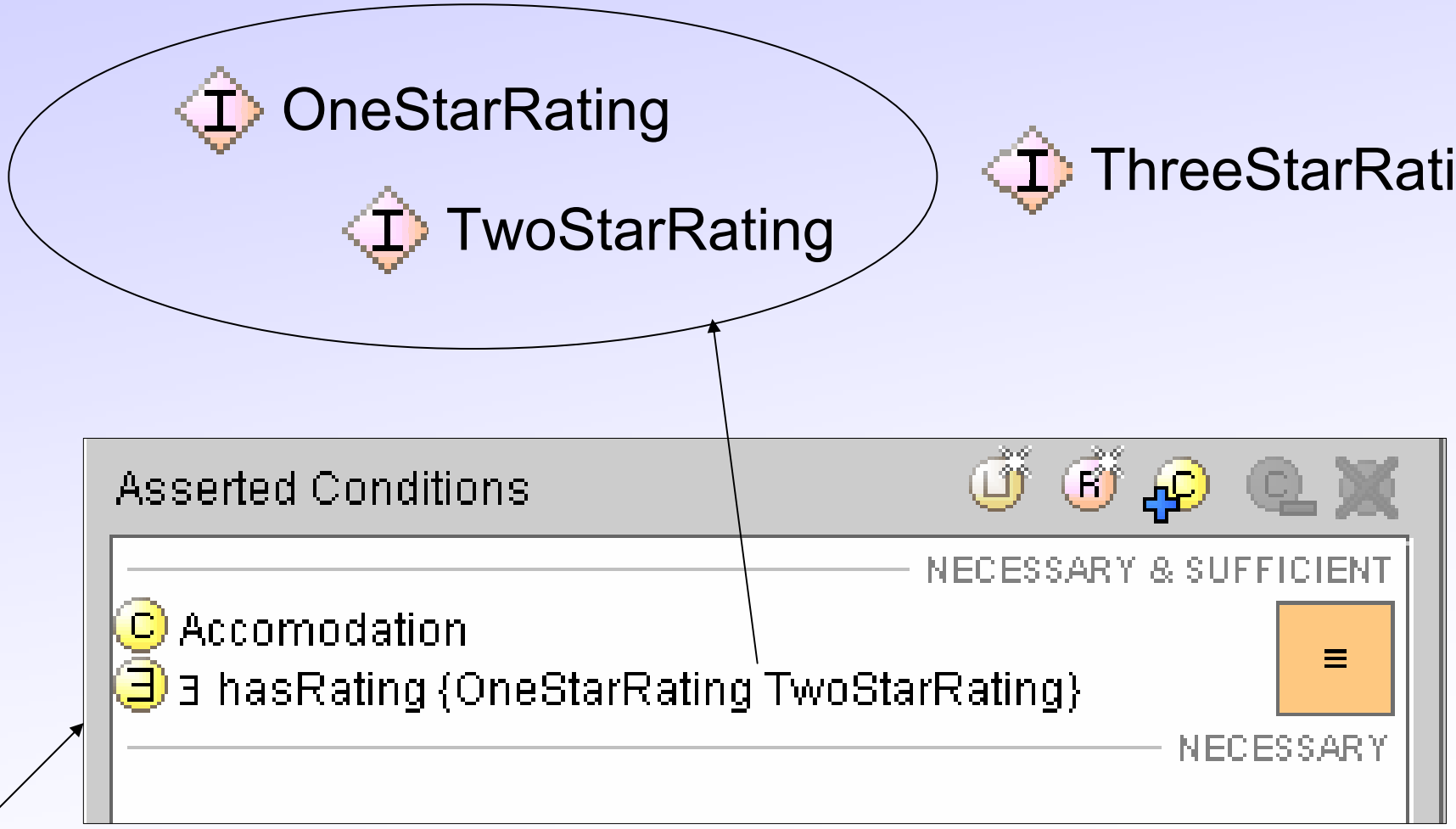
hasValue Restrictions

- Meaning: At least one of the values of the property is a certain value
- Similar to someValuesFrom ∃ but with Individuals and primitive values
- Example: A PartOfSydney is a Destination where one of the values of the isPartOf property is Sydney



Enumerated Classes

- Consist of exactly the listed individuals

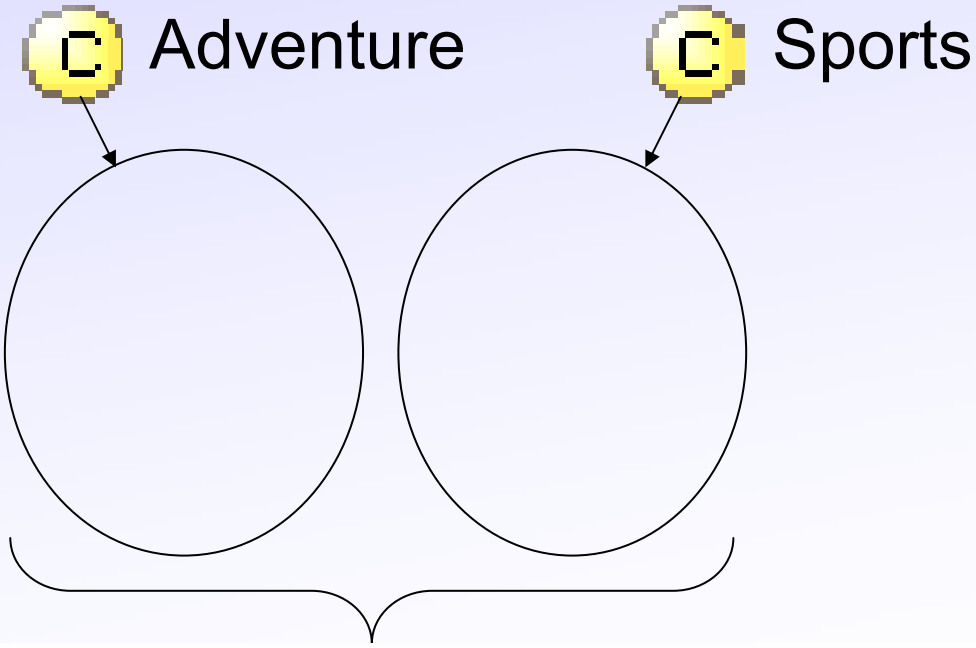


Logical Class Definitions

- Define classes out of other classes
  -  unionOf (or)
  -  intersectionOf (and)
  -  complementOf (not)
- Allow arbitrary nesting of class descriptions (A and (B or C) and not D)

unionOf

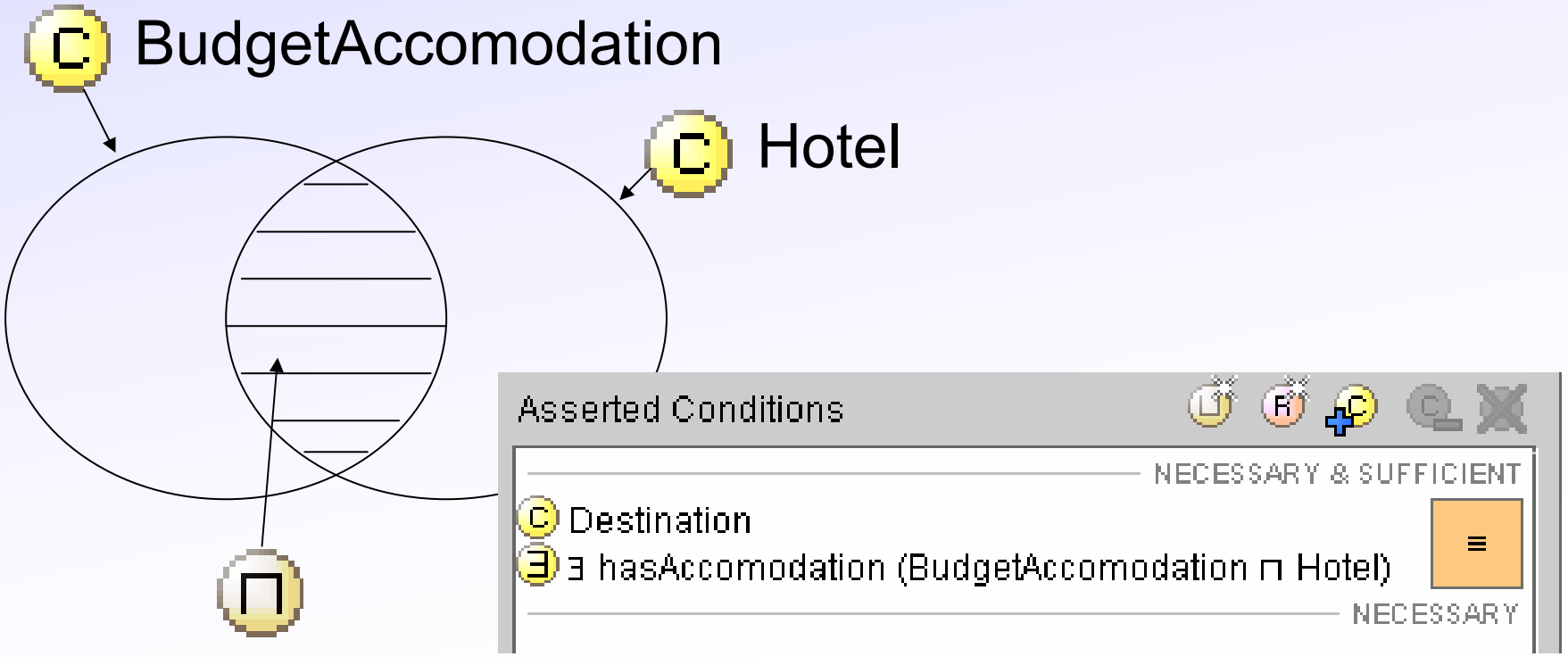
- The class of individuals that belong to class A **or** class B (or both)
- Example: Adventure or Sports activities



$\exists$ $\exists$ hasActivity (Sports $\sqcup$ Adventure)

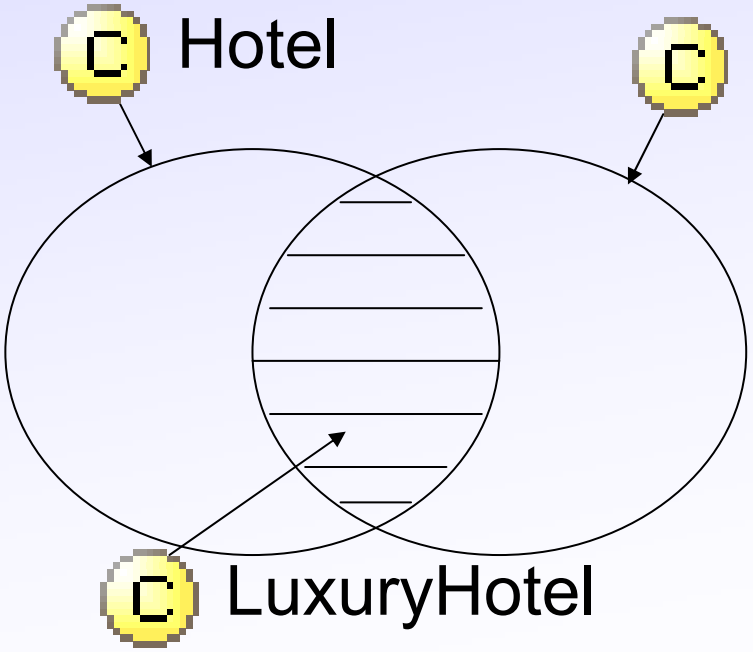
intersectionOf

- The class of individuals that belong to both class A **and** class B
- Example: A BudgetHotelDestination is a destination with accomodation that is a budget accomodation **and** a hotel



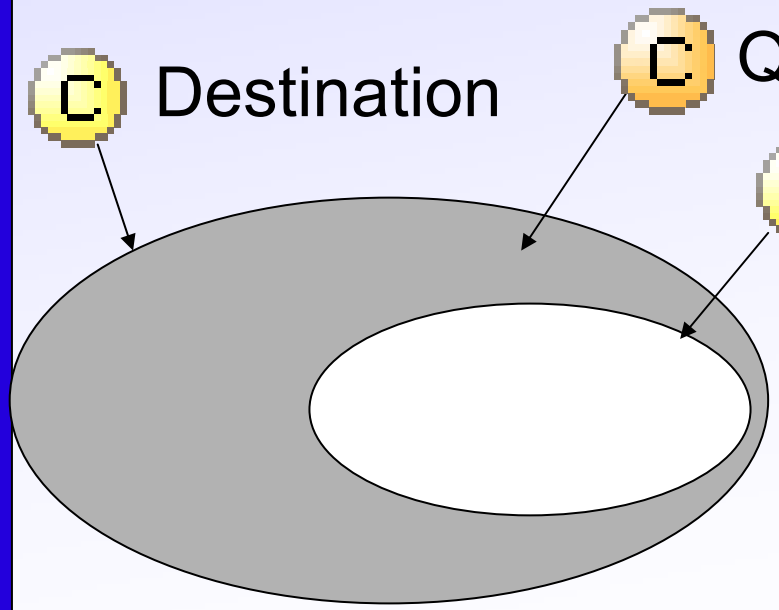
Implicit intersectionOf

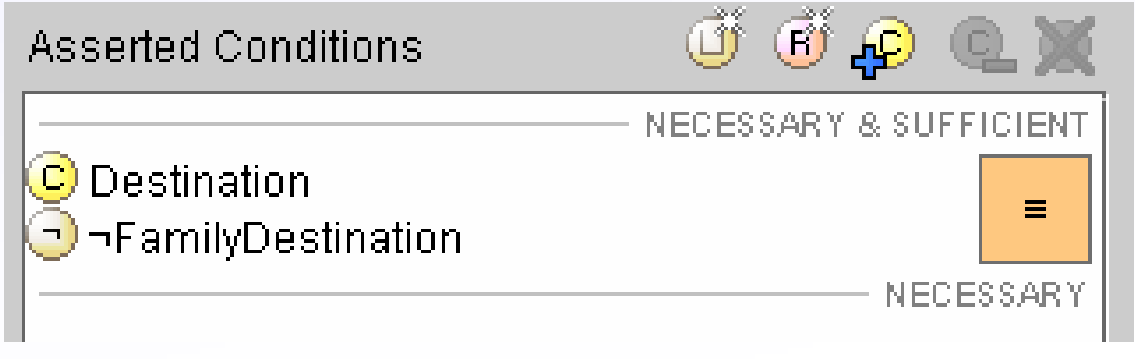
- When a class is defined by more than one class description, then it consists of the intersection of the descriptions
- Example: A luxury hotel is a hotel that is also an accommodation with 3 stars



complementOf

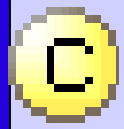
- The class of all individuals that do not belong to a certain class
- Example: A quiet destination is a destination that is **not** a family destination





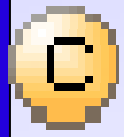
Class Conditions

- **Necessary Conditions**:



(Primitive / partial classes)
"If we know that something is a X,
then it must fulfill the conditions..."

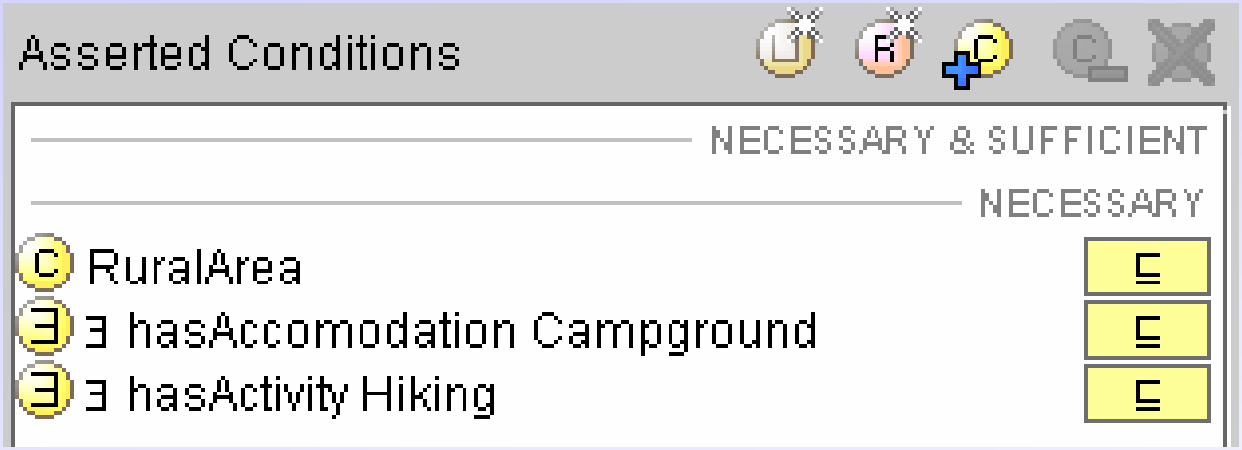
- **Necessary & Sufficient Conditions**:



(Defined / complete classes)
"If something fulfills the conditions...,
then it is an X."

Class Conditions (2)

NationalPark ⟶



(not everything that fulfills these conditions is a NationalPark)

QuietDestination ⟷

(everything that fulfills these conditions is a QuietDestination)

Classification

NationalPark

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

RuralArea	⊑
∃ hasAccomodation Campground	⊑
∃ hasActivity Hiking	⊑

- A RuralArea is a Destination
- A Campground is BudgetAccomodation
- Hiking is a Sport
- Therefore:
  Every NationalPark is a Backpackers-Destination

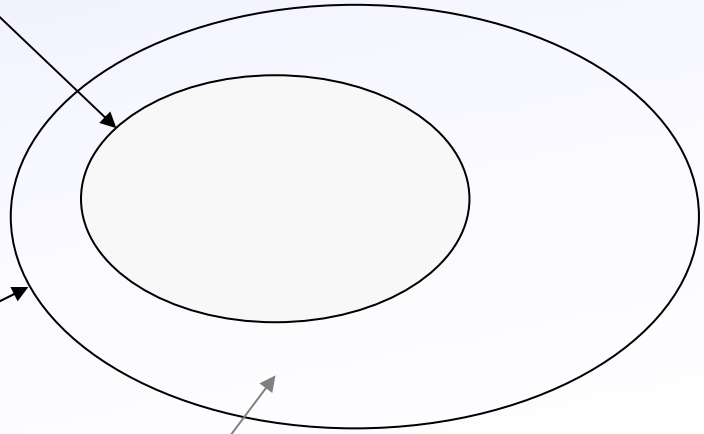
BackpackersDestination

Asserted Conditions

NECESSARY & SUFFICIENT

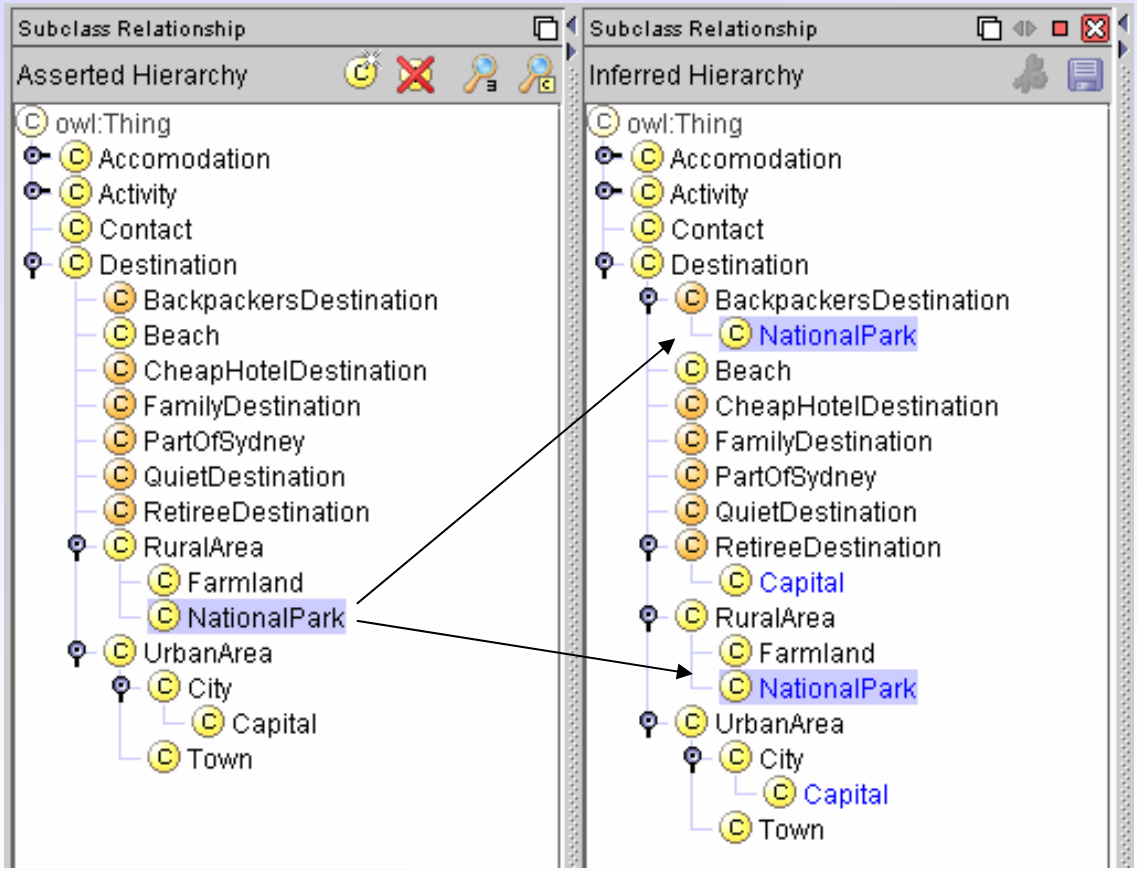
Destination	≡
∃ hasAccomodation BudgetAccomodation	
∃ hasActivity (Sports ⊔ Adventure)	

NECESSARY

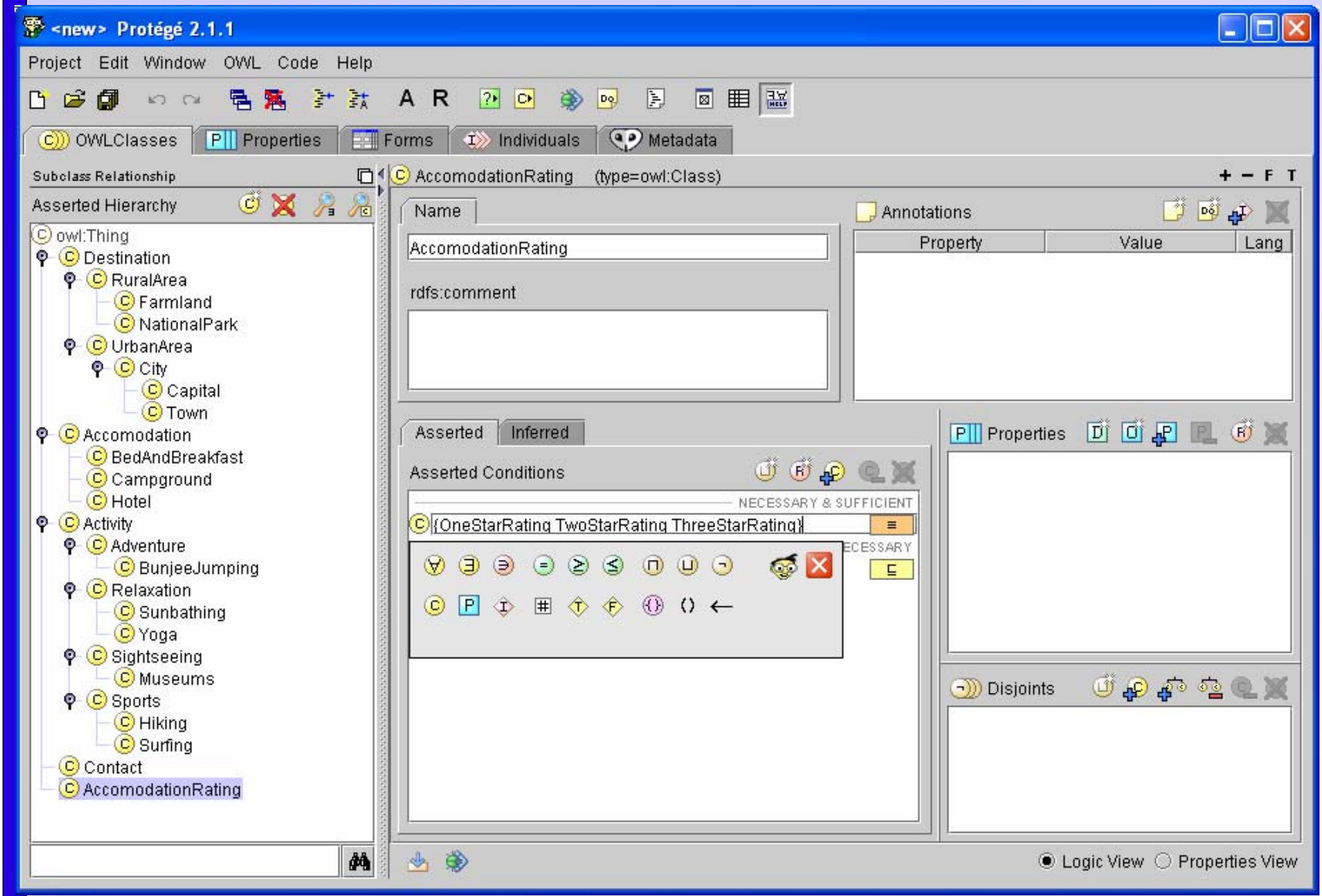


Classification (2)

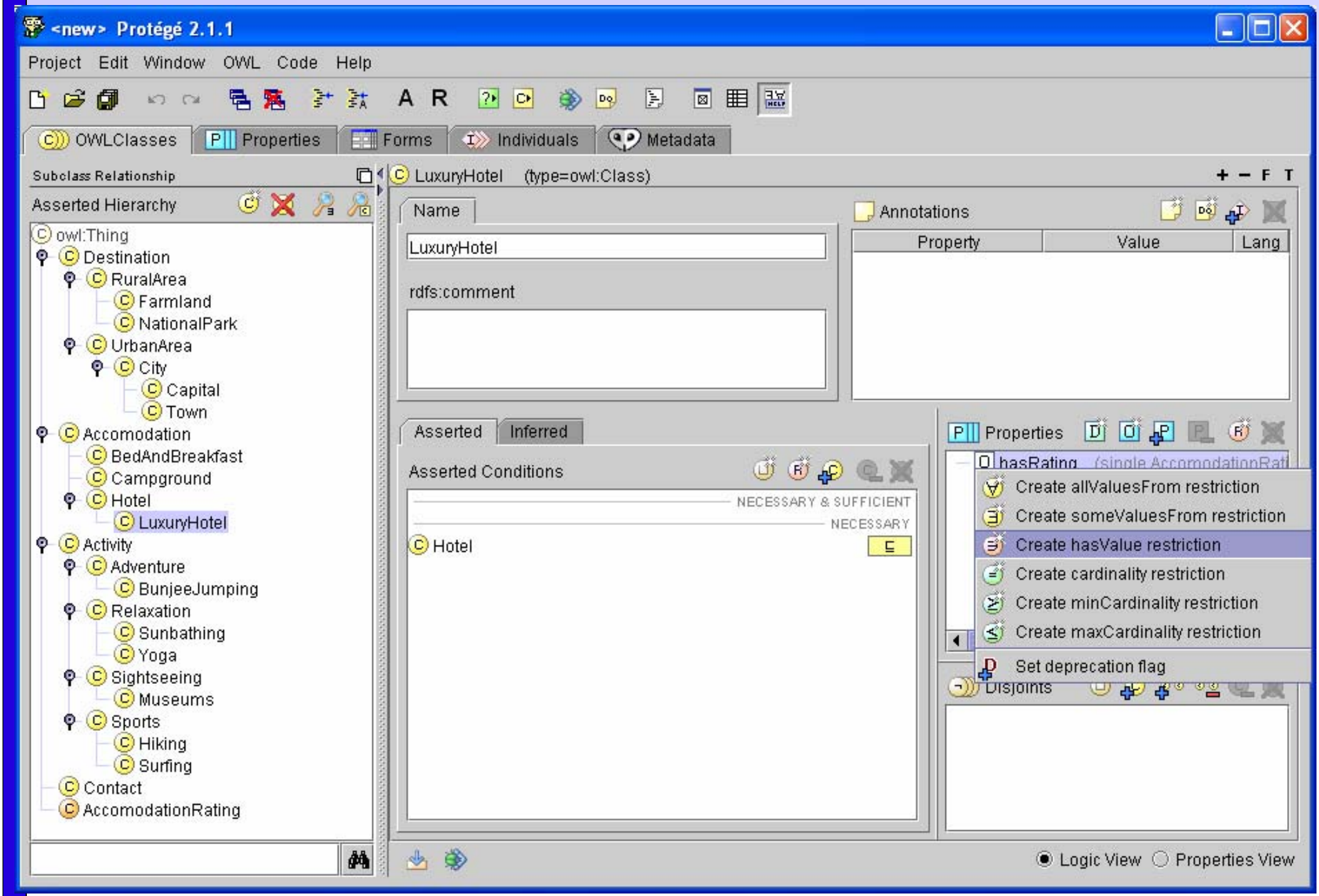
- Input: Asserted class definitions
- Output: Inferred subclass relationships



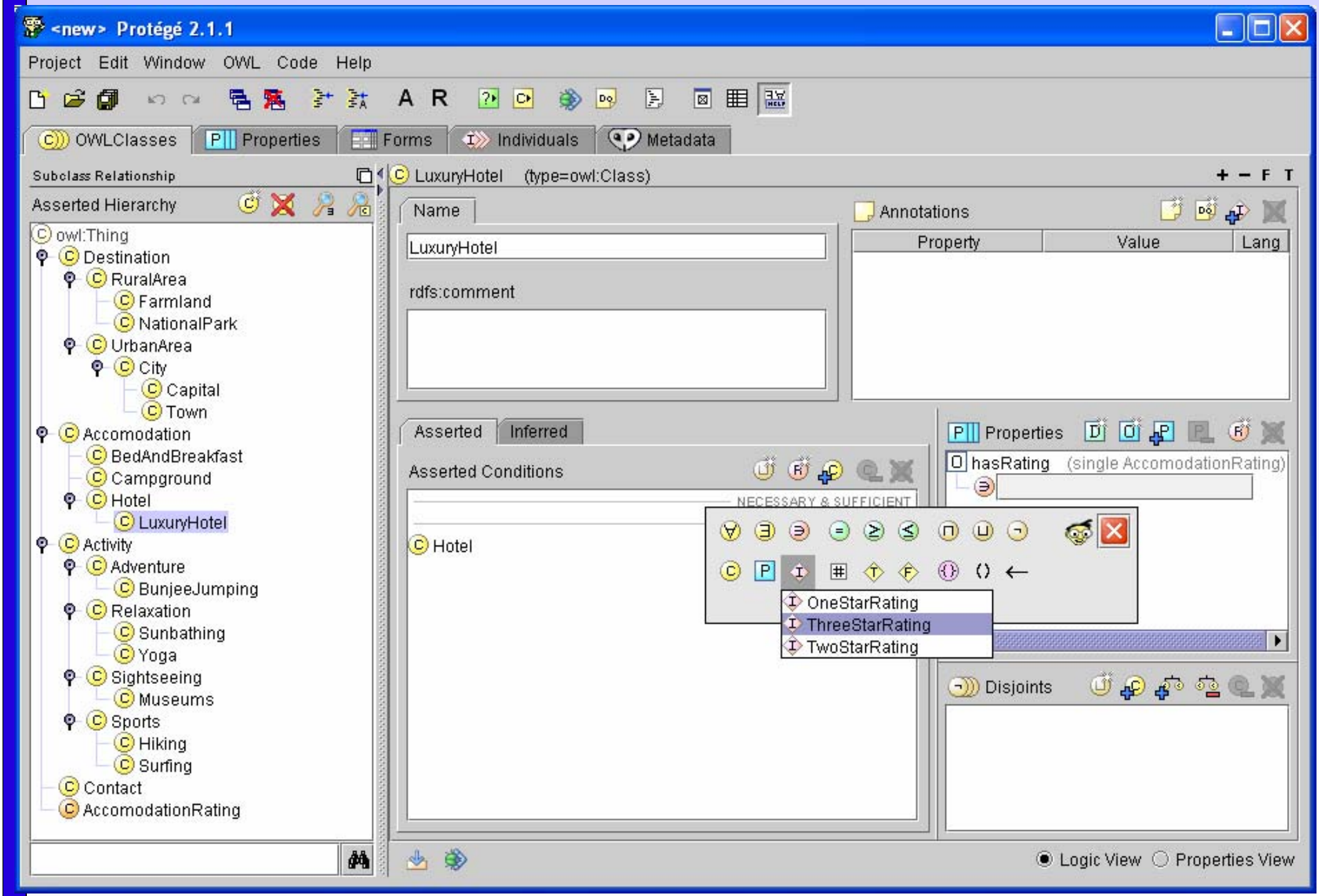
(Create an enumerated class out of individuals)



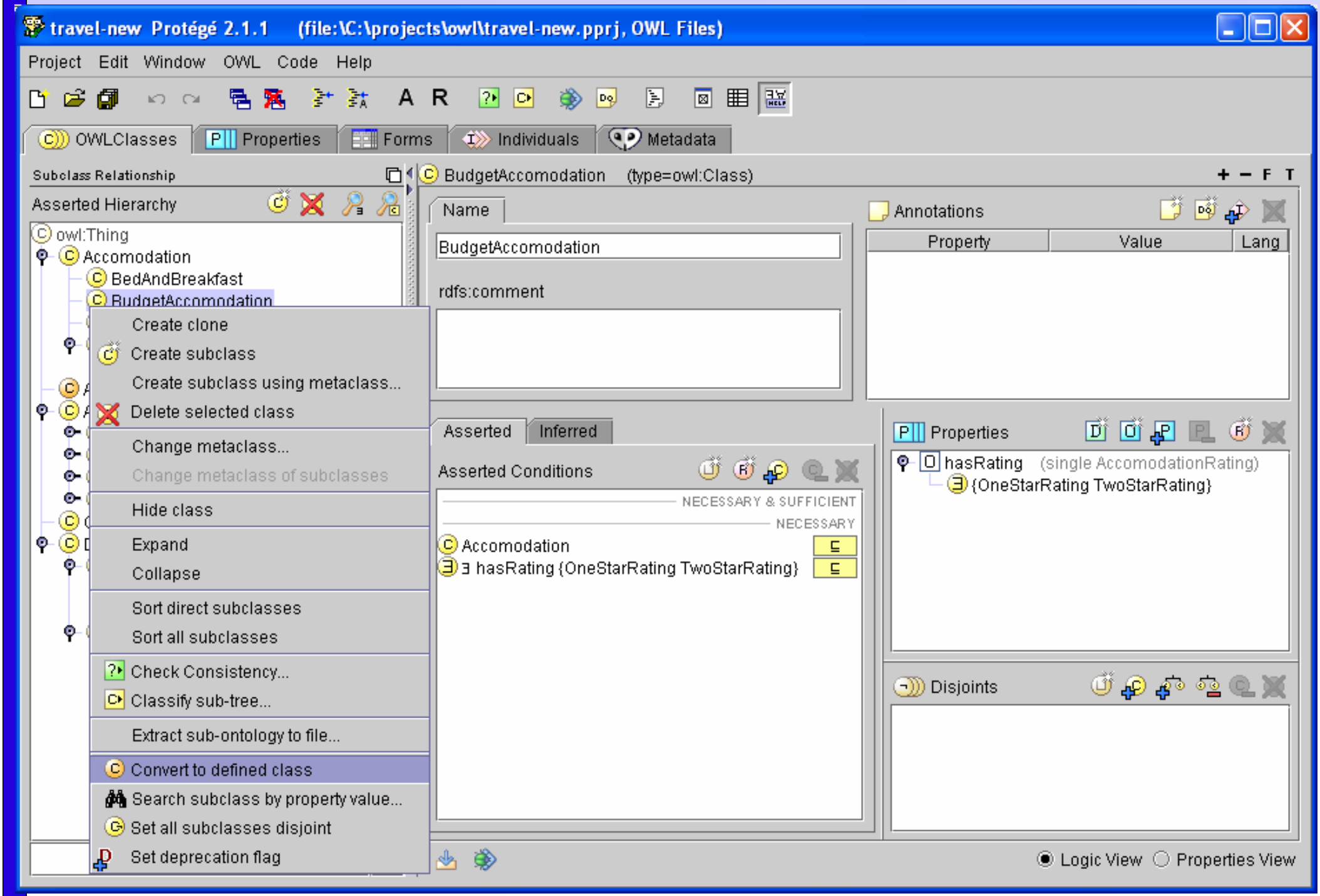
(Create a hasValue restriction)



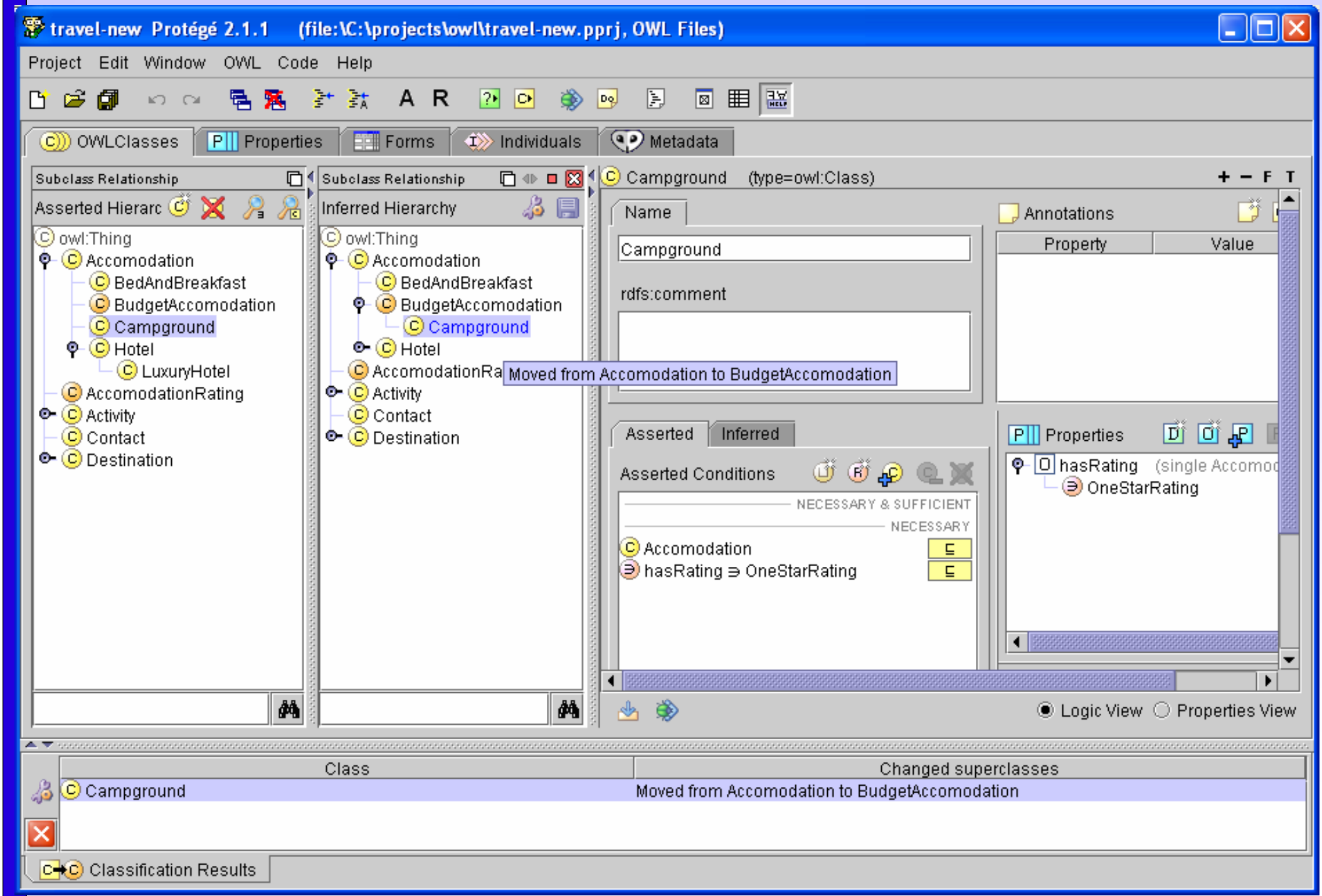
(Create a hasValue restriction)



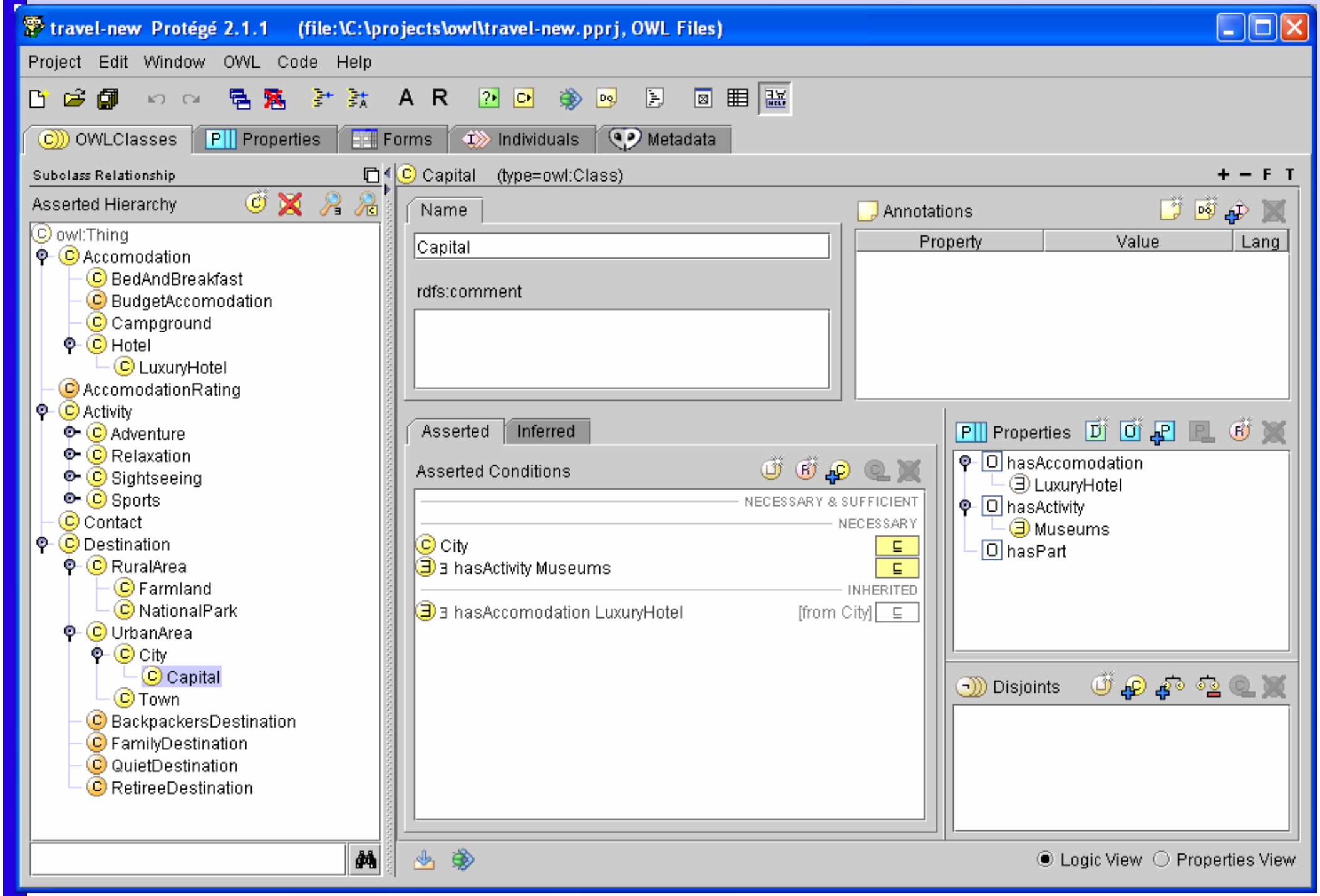
(Create a defined class)



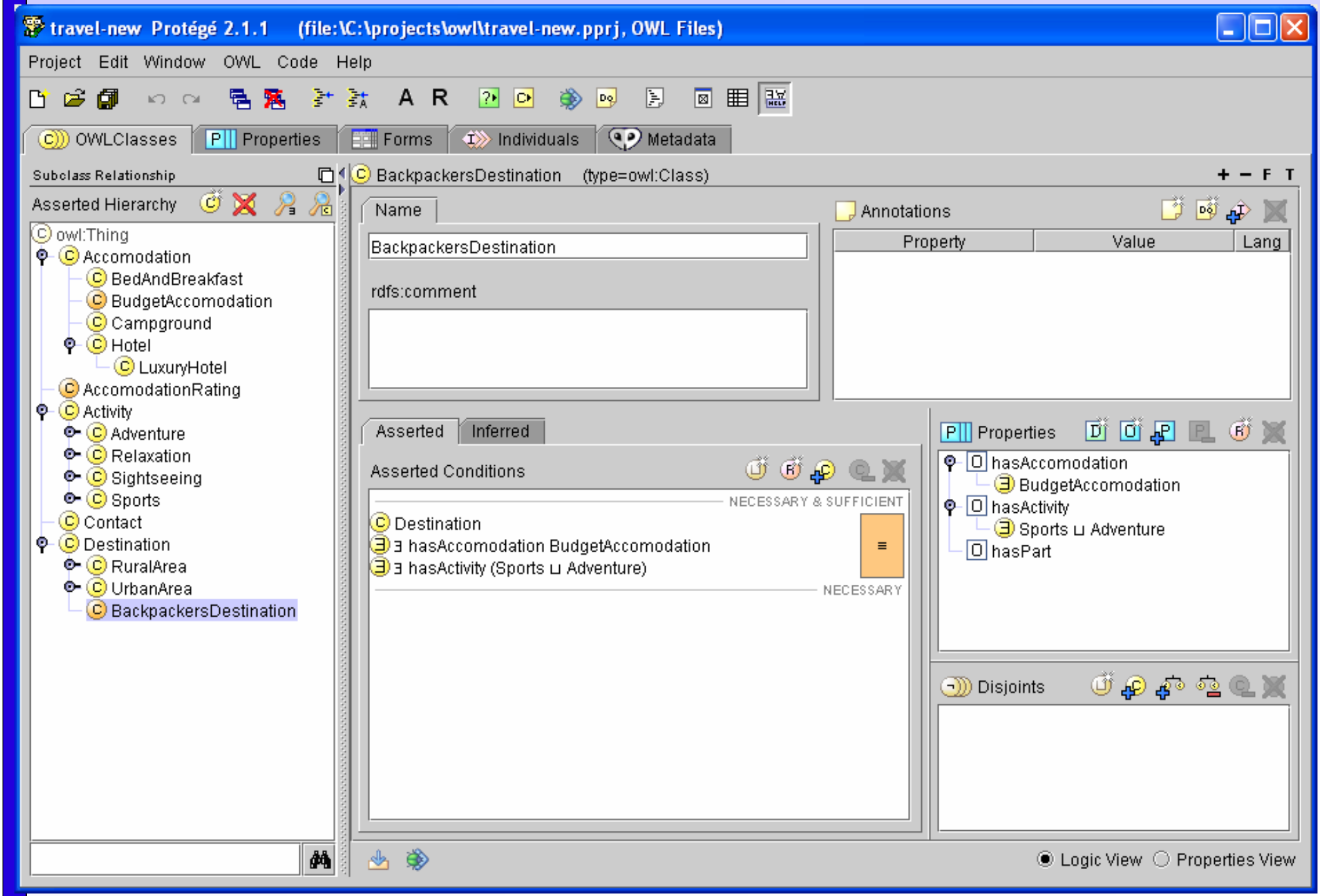
(Classify Campground)



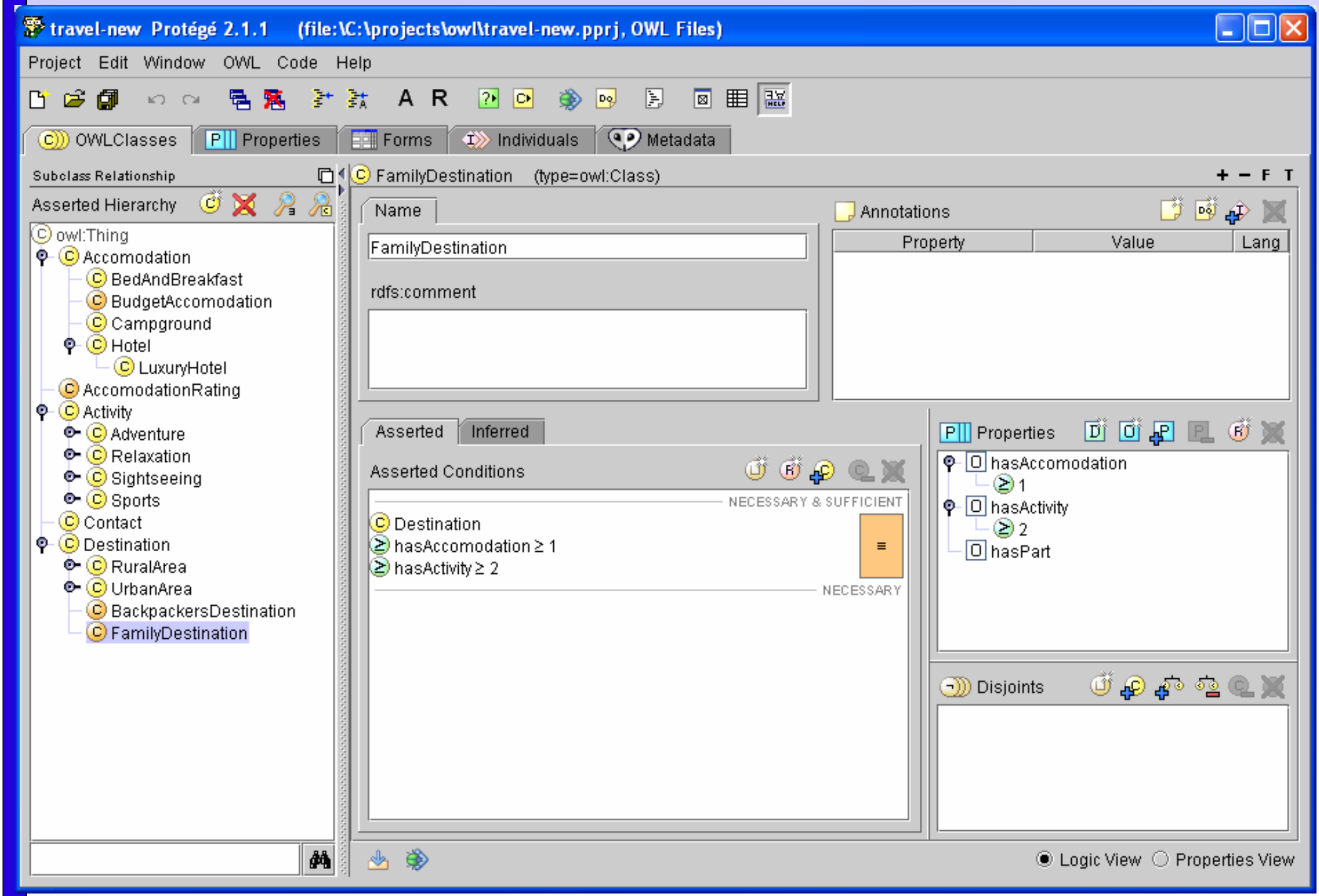
(Add restrictions to City and Capital)



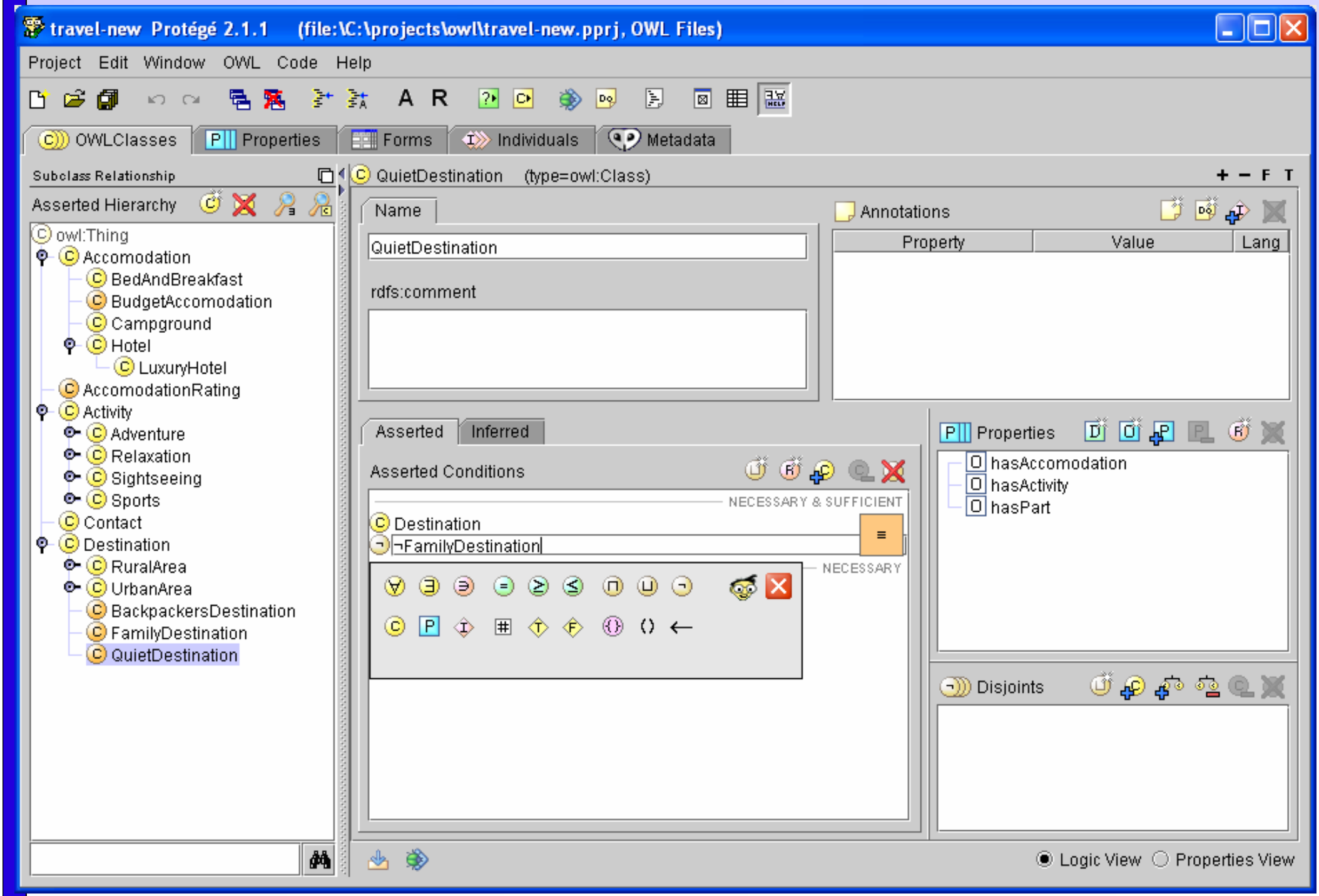
(Create defined class BackpackersDestination)



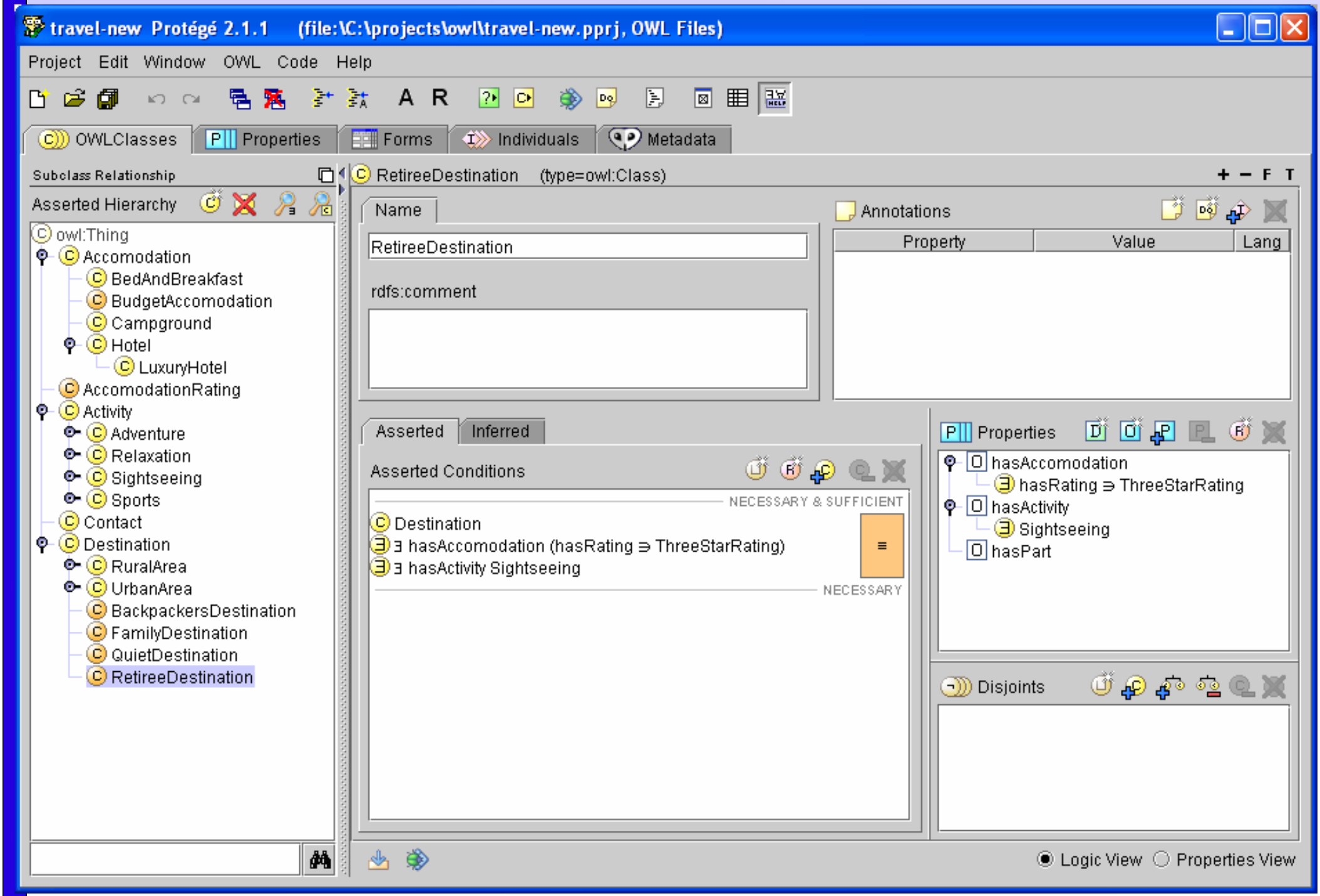
(Create defined class FamilyDestination)



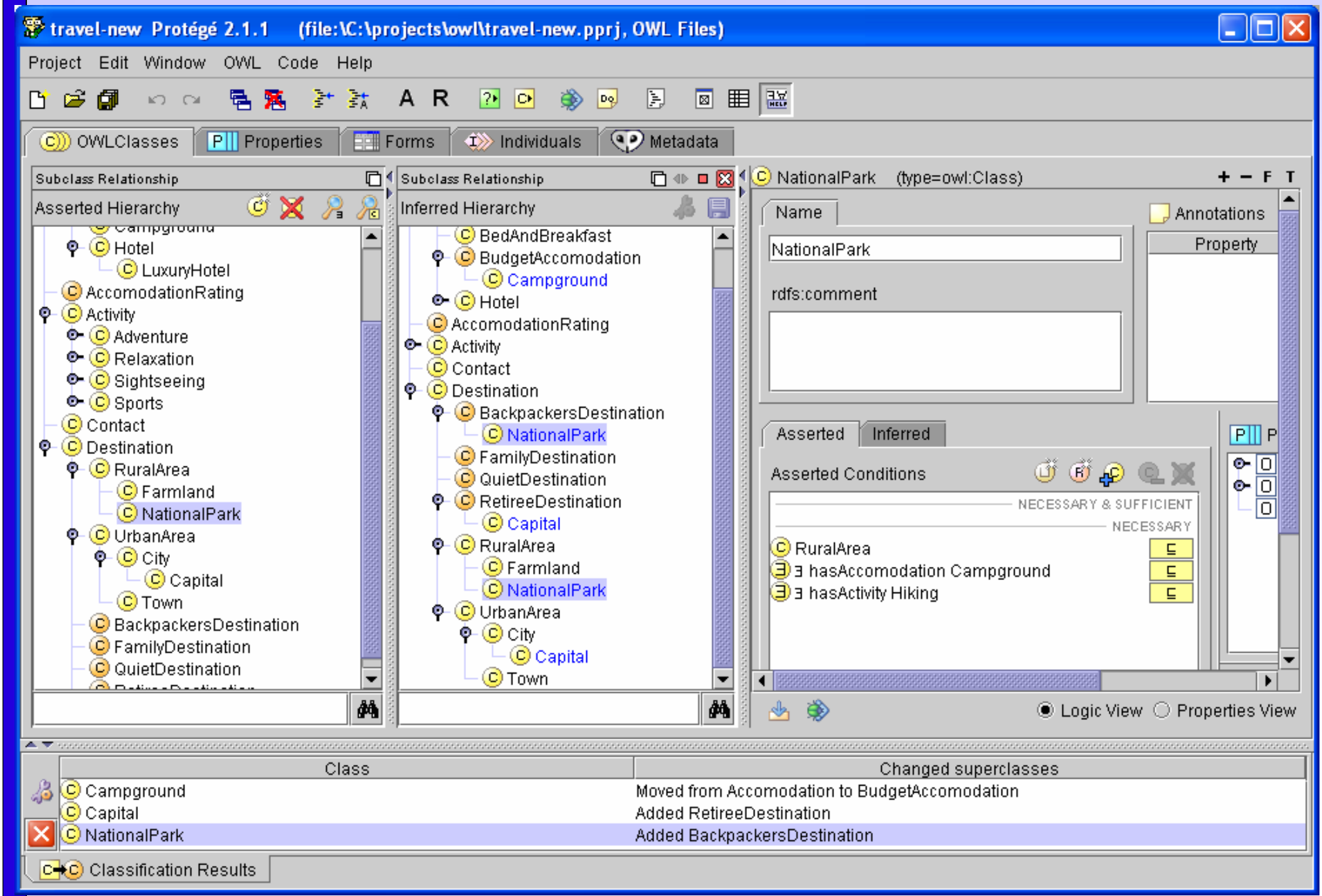
(Create defined class QuietDestination)



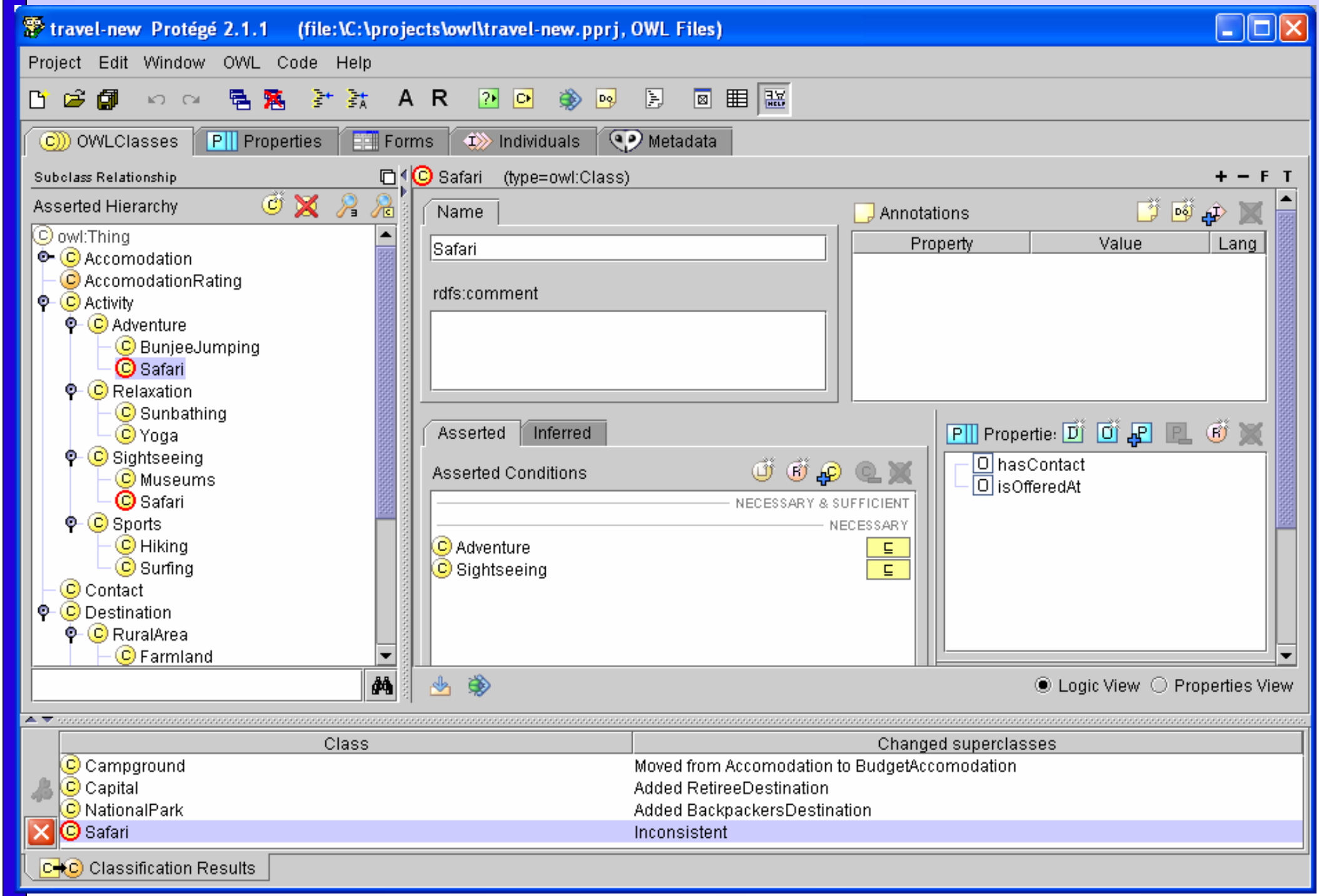
(Create defined class RetireeDestination)



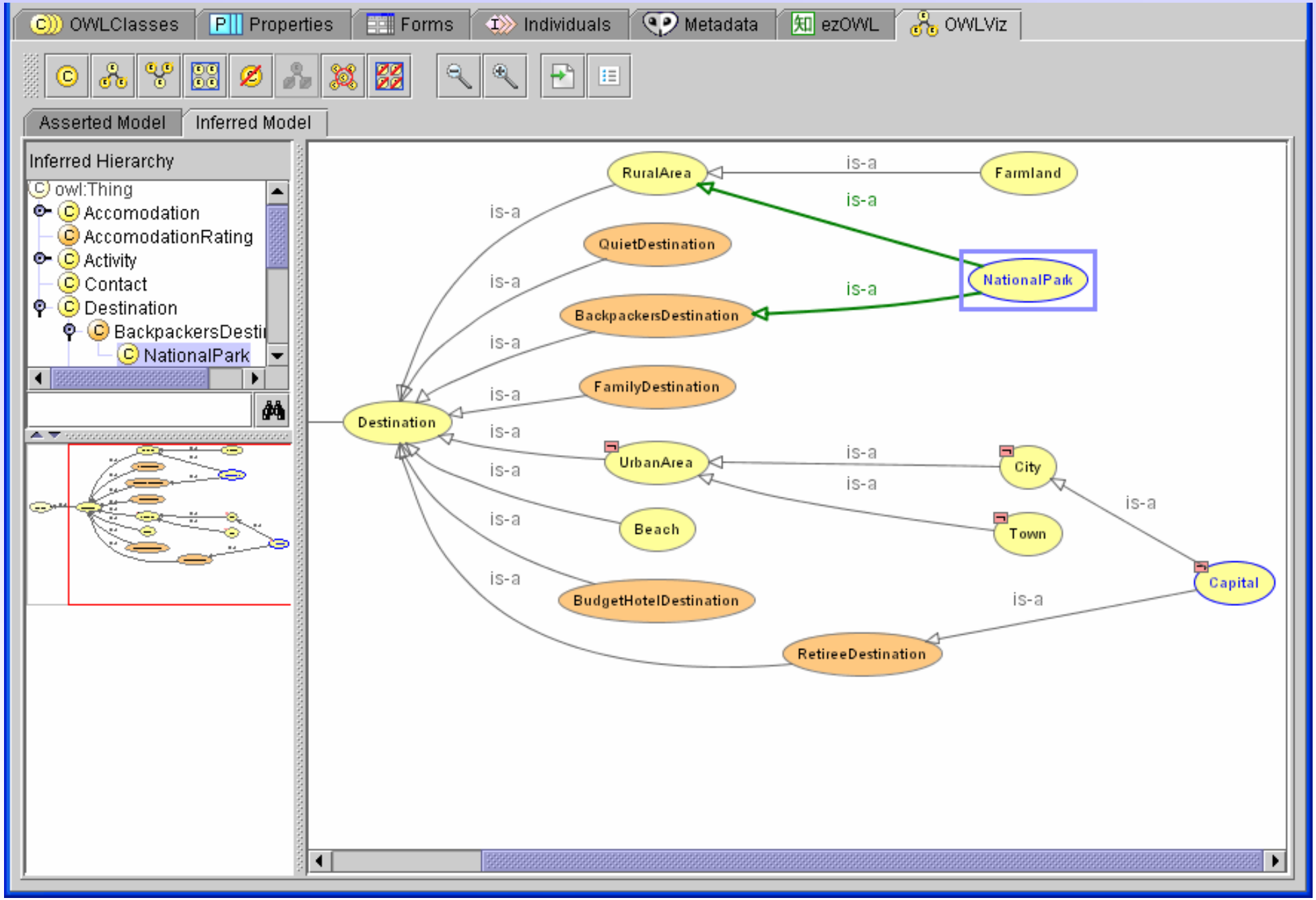
(Classification)



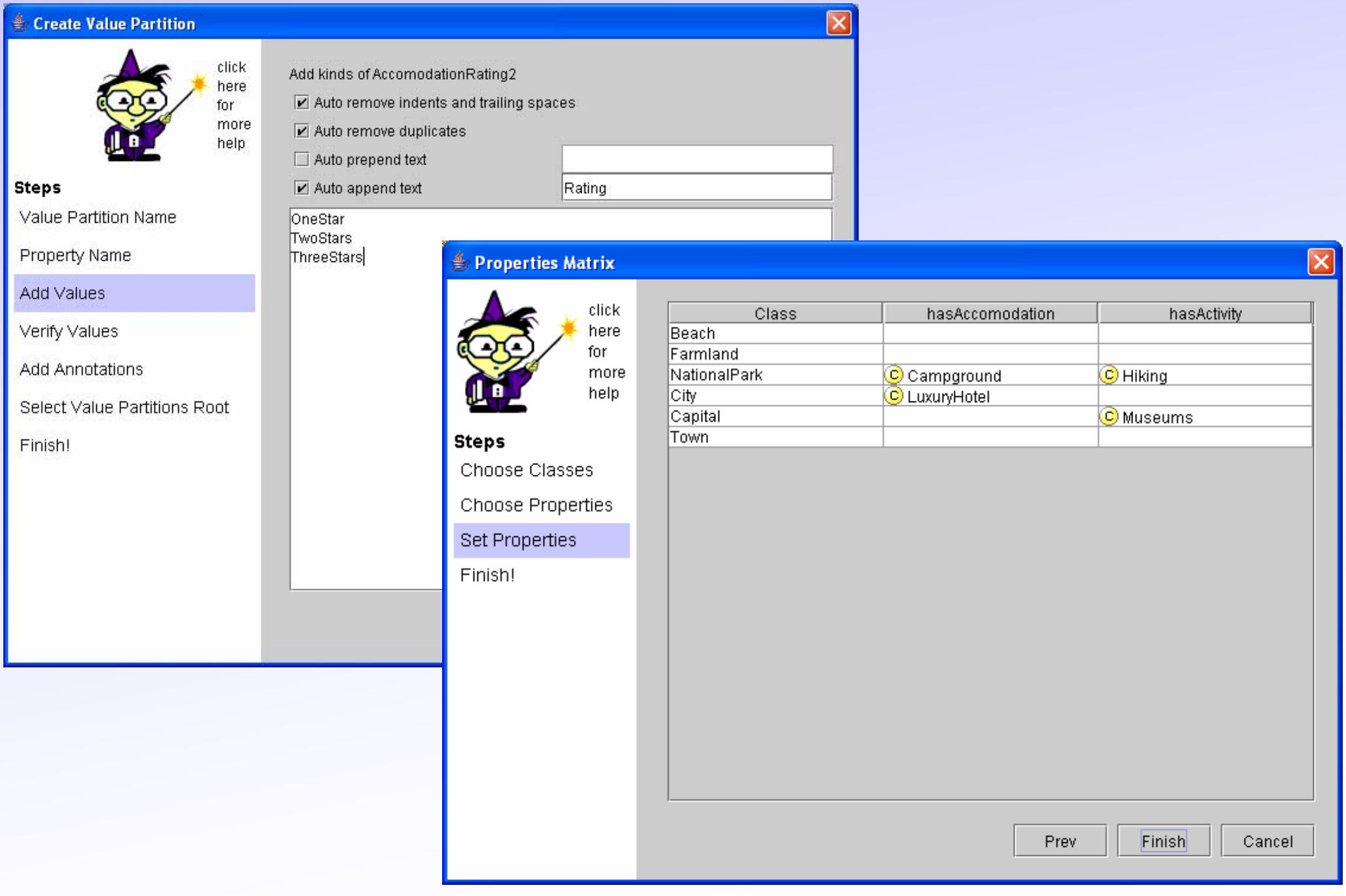
(Consistency Checking)



Visualization with OWLViz



OWL Wizards


Create Value Partition
click here for more help
Steps
Value Partition Name
Property Name
Add Values
Verify Values
Add Annotations
Select Value Partitions Root
Finish!
Add kinds of AccomodationRating2
Auto remove indents and trailing spaces
Auto remove duplicates
Auto prepend text
Auto append text
Rating
OneStar
TwoStars
ThreeStars
Properties Matrix
click here for more help
Steps
Choose Classes
Choose Properties
Set Properties
Finish!
Class
hasAccomodation
hasActivity
Beach
Farmland
NationalPark
Campground
Hiking
City
LuxuryHotel
Capital
Museums
Town
Prev
Finish
Cancel

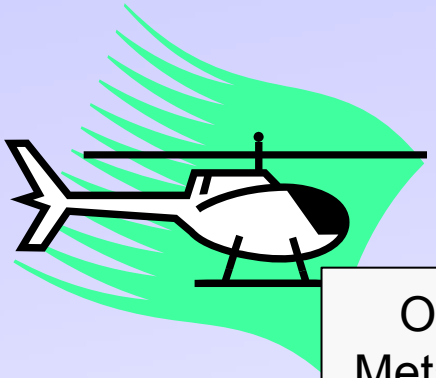
Putting it All Together

- Ontology has been developed
- Published on a dedicated web address
- Ontology provides standard terminology
- Other ontologies can extend it
- Users can instantiate the ontology to provide instances
  - specific hotels
  - specific activities

Ontology Import

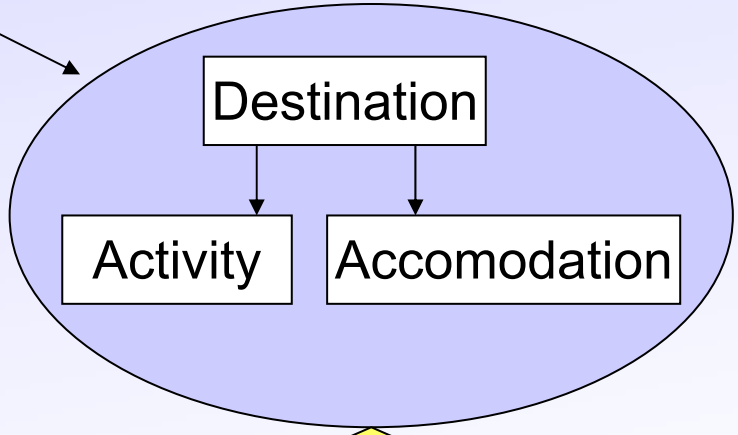
- Adds all classes, properties and individuals from an external OWL ontology into your project
- Allows to create individuals, subclasses, or to further restrict imported classes
- Can be used to instantiate an ontology for the Semantic Web

Tourism Semantic Web (2)

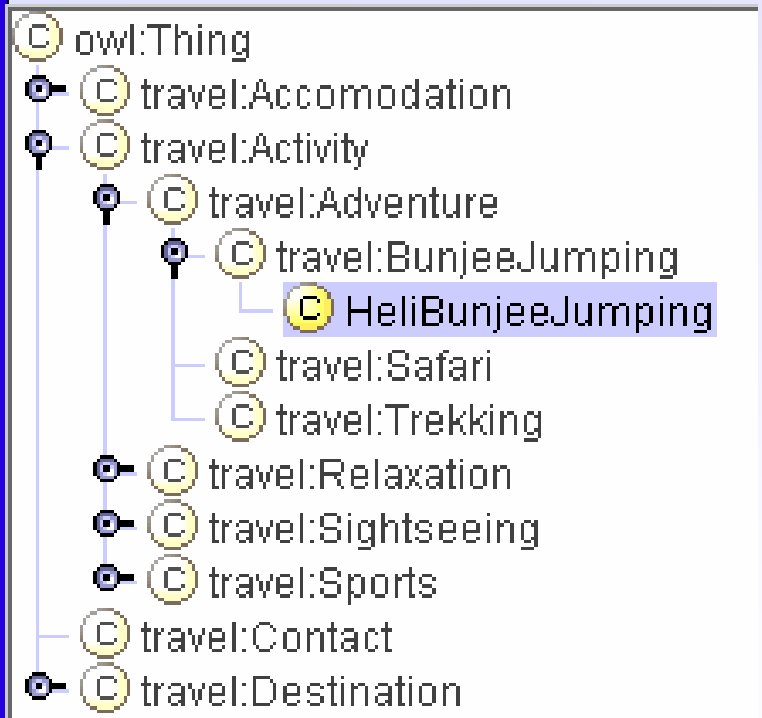


OWL Metadata (Individuals)

Tourism Ontology

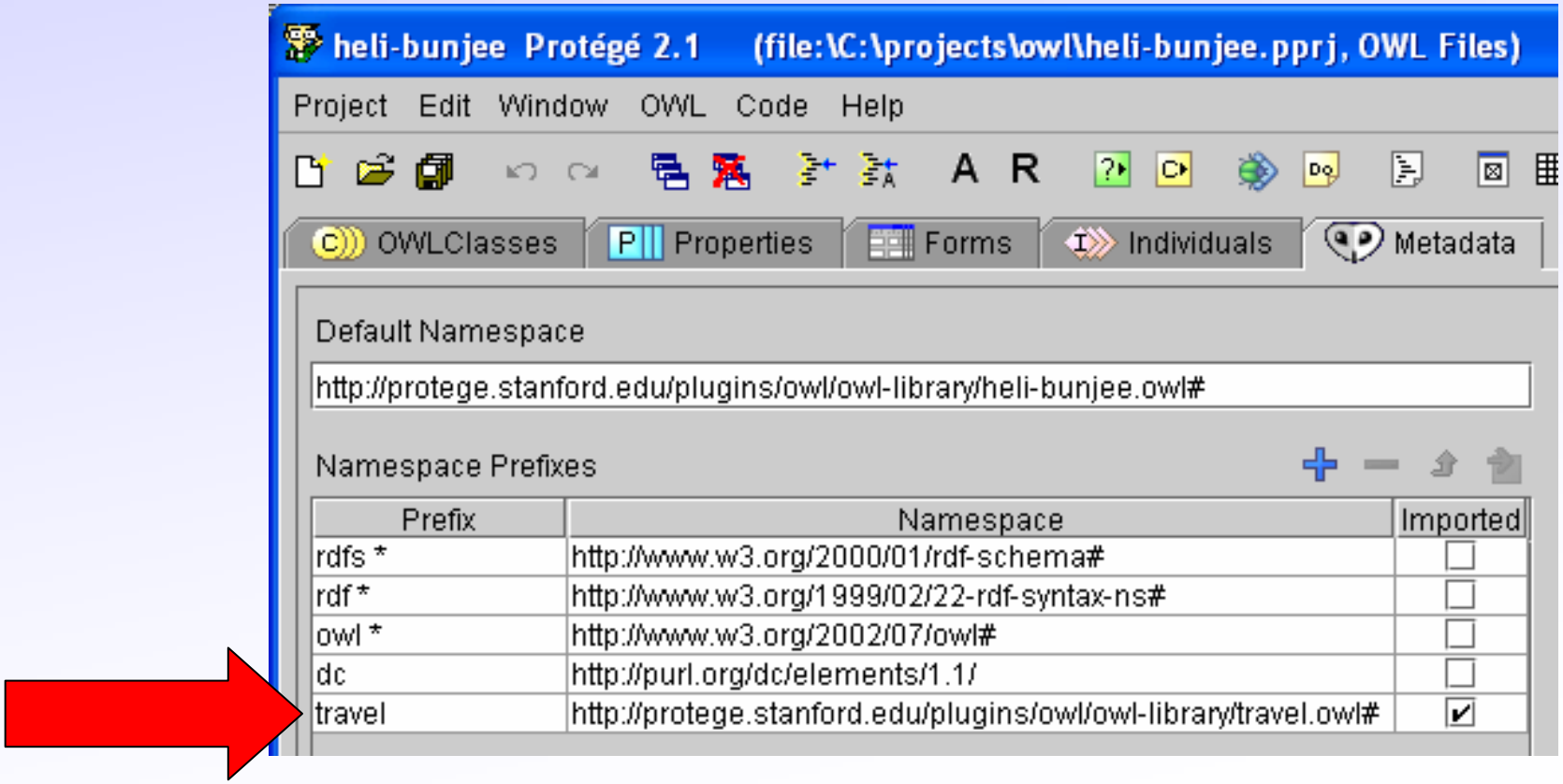


Web Services

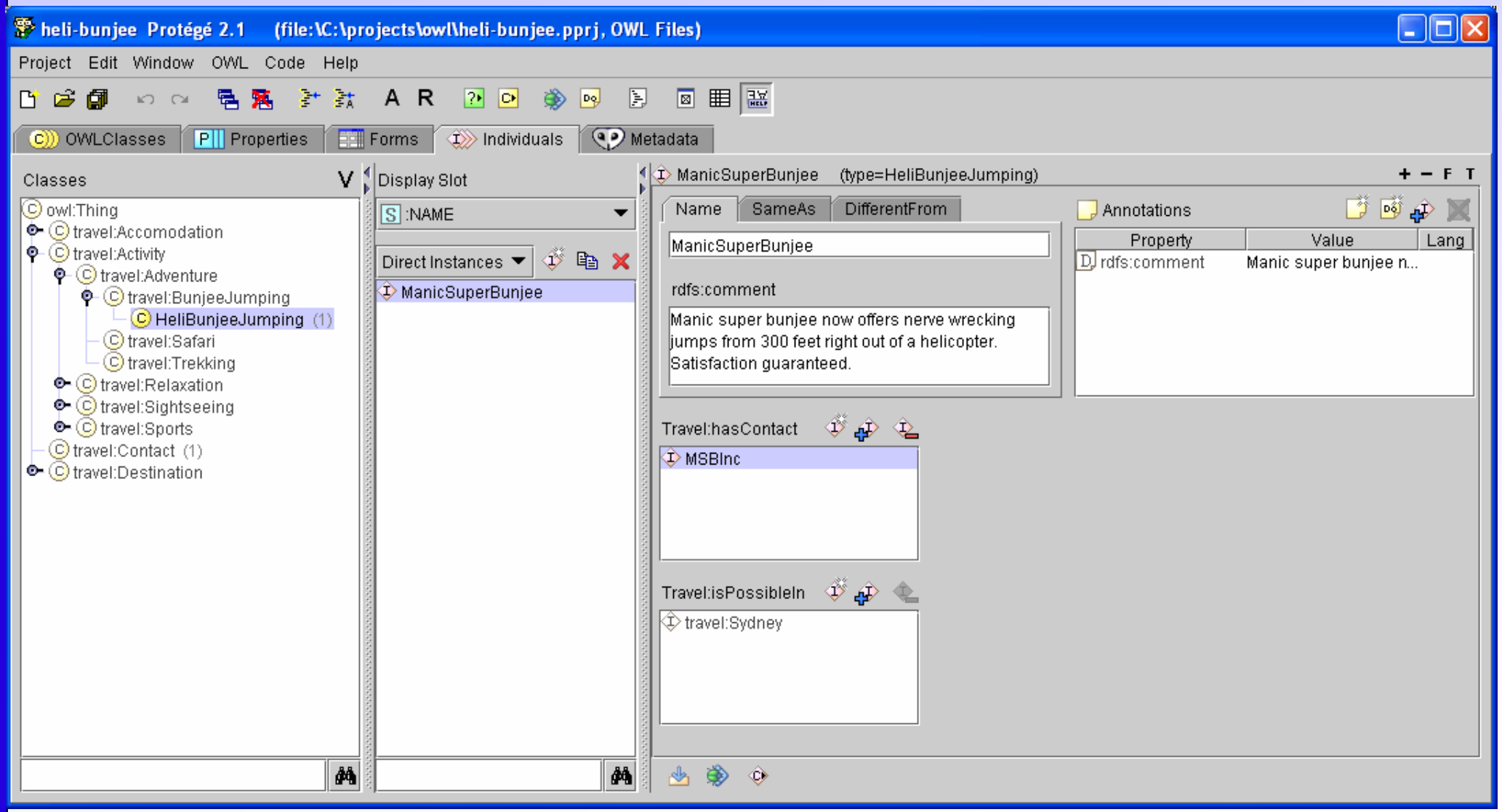


Ontology Import with Protégé

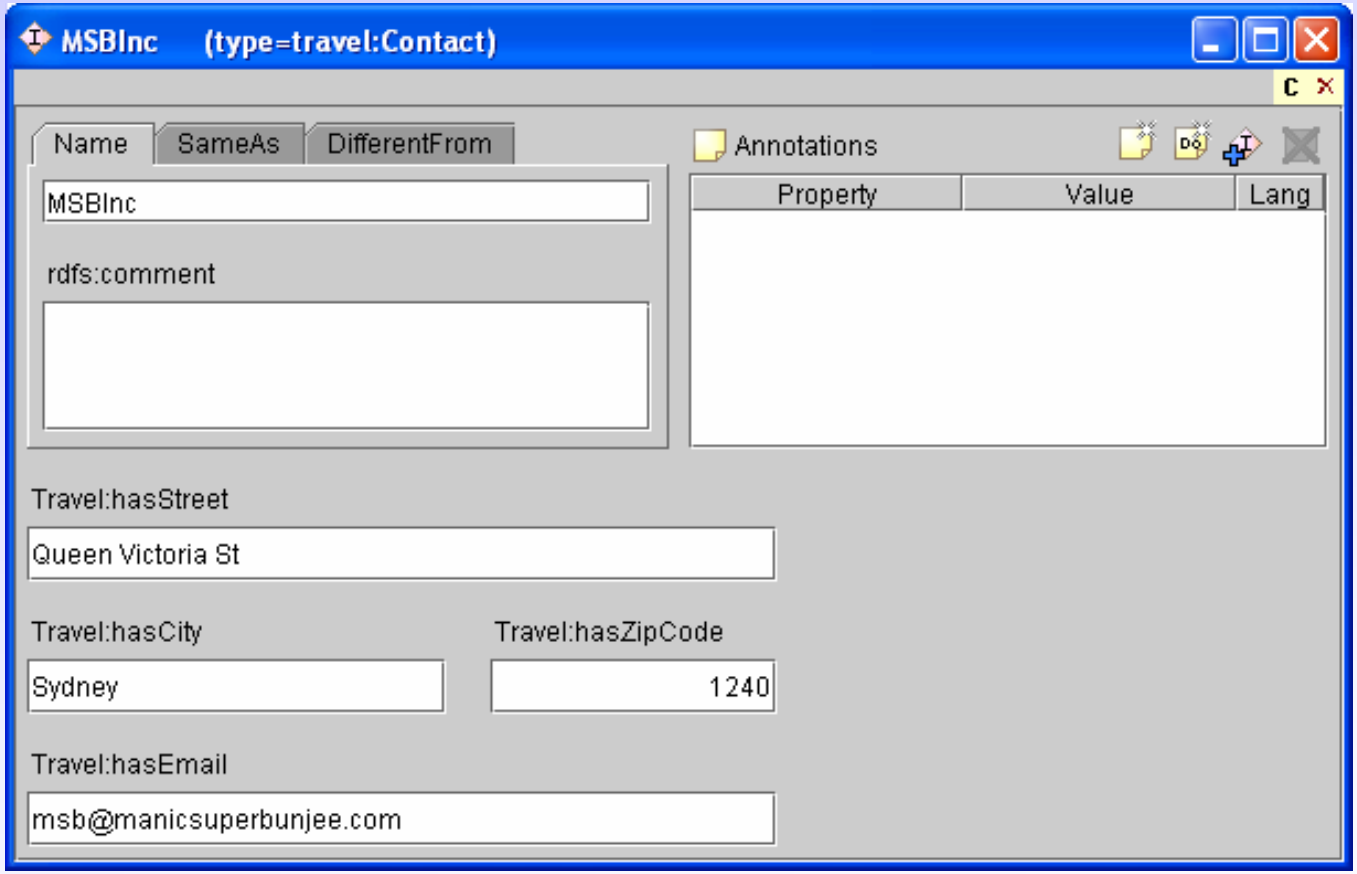
- On the Metadata tab:
  - Add namespace, define prefix
  - Check “Imported” and reload your project



Individuals



Individuals



OWL File

```
<?xml version="1.0"?>\
<rdf:RDF
    xmlns="http://protege.stanford.edu/plugins/owl/owl-library/heli-bunjee.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:travel="http://protege.stanford.edu/plugins/owl/owl-library/travel.owl#"
  xml:base="http://protege.stanford.edu/plugins/owl/owl-library/heli-bunjee.owl">

  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/owl-library/travel.owl"/>
  </owl:Ontology>

  <owl:Class rdf:ID="HeliBunjeeJumping">
    <rdfs:subClassOf rdf:resource="http://protege.stanford.edu/plugins/owl/owl-library/travel.owl#BunjeeJumping"/>
  </owl:Class>

  <HeliBunjeeJumping rdf:ID="ManicSuperBunjee">
    <travel:isPossibleIn>
      <rdf:Description rdf:about="http://protege.stanford.edu/plugins/owl/owl-library/travel.owl#Sydney">
        <travel:hasActivity rdf:resource="#ManicSuperBunjee"/>
      </rdf:Description>
    </travel:isPossibleIn>
    <travel:hasContact>
      <travel:Contact rdf:ID="MSBInc">
        <travel:hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string">msb@manicsuperbunjee.com
        </travel:hasEmail>
        <travel:hasCity rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sydney</travel:hasCity>
        <travel:hasStreet rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Queen Victoria St</travel:hasStreet>
        <travel:hasZipCode rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1240</travel:hasZipCode>
      </travel:Contact>
    </travel:hasContact>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Manic super bunjee now offers nerve
        wrecking jumps from 300 feet right out of a helicopter.  Satisfaction guaranteed.</rdfs:comment>
  </HeliBunjeeJumping>

</rdf:RDF>
```