Πανεπιστήμιο Μακεδονίας
Τμ. Εφαρμοσμένης Πληροφορικής

ECLipSe Constraints on Sets

Δομή

- Βιβλιοθήκη ic_sets
- Παραδείγματα

Η βιβλιοθήκη ic_sets (1/2)

- Η βιβλιοθήκη ic_sets χειρίζεται σύνολα ακεραίων.
- Ένα σύνολο ακεραίων είναι μια *διατεταγμένη λίστα* ακεραίων αριθμών.
  - S1 = [1, 3, 7]
  - S2= [ ]
- Ορίζουμε μια μεταβλητή συνόλου δίνοντας τα στοιχεία που *σίγουρα* ανήκουν στο σύνολο και τα στοιχεία που *πιθανά* να ανήκουν στο σύνολο:
  - ?- S1 :: [2, 3] .. [1, 2, 3, 4].

Η βιβλιοθήκη ic_sets (2/2)

- ?- S1 :: [2, 3] .. [1, 2, 3, 4].
  - S1 = S1{[2, 3] \/ ([] .. [1, 4]) : _313{2 .. 4}}
  - Η παραπάνω δήλωση σημαίνει ότι η μεταβλητή S1 είναι ένα σύνολο το οποίο περιέχει σίγουρα τα στοιχεία 2 και 3, ενώ μπορεί ενδεχομένως να περιέχει και τα 1 και 4.
  - Στην απάντησή της η ECLiPSe μας δίνει και τον **πληθικό αριθμό** του συνόλου.
- **Προσοχή**: Εάν έχουμε φορτώσει και τη βιβλιοθήκη ic, η παραπάνω δήλωση θα έπρεπε να γραφεί ως:
  - ic_sets:(S1 :: [2, 3]..[1,2,3,4])

Εναλλακτικοί τρόποι ορισμού συνόλων

- Η δήλωση
  - S1 :: [] .. [1, 2, 3, 4].
- μπορεί εναλλακτικά να γραφεί
  - S1 subset [1,2,3,4]
  - intset(S1,1,4)
- intsets(?Sets, ?N, +Min, +Max)
  - Sets λίστα από N sets καθένα από τα οποία είναι ανήκει στο **[]...[Min,Min+1,...,Max]**

Πρόσβαση στο πεδίο

- potential_members(?Set, -List)
  - -List, πιθανά μέλη του συνόλου, δηλ. εκείνα για τα οποία δεν είναι "σίγουρο" ότι ακήκουν στο σύνολο.
- set_range(?Set, -Lwb, -Upb)
  - το έλάχιστο και το μέγιστο σύνολο (όρια) του συνόλου ?Set

Περιορισμοί συνόλων

- Μπορούμε να θέσουμε περιορισμούς μεταξύ συνόλων ή μεταξύ ακεραίων και συνόλων.
  - ?X in ?Set : Ο ακέραιος X **είναι μέλος** του συνόλου Set
  - ?X notin ?Set : Ο ακέραιος X **δεν είναι μέλος** του συνόλου Set
  - #(?Set, ?Card) : Ο πληθικός αριθμός του συνόλου Set είναι Card

Περιορισμοί συνόλων

- ?Set1 disjoint ?Set2 : Τα Set1 και Set2 είναι ξένα.
- ?Set1 sameset ?Set2 : Τα σύνολα Set1 και Set2 είναι ίδια.
- ?Set1 includes ?Set2 : Το σύνολο Set1 είναι υπερσύνολο του Set2.
- ?Set1 subset ?Set2 : Το σύνολο Set1 είναι υποσύνολο του Set2.
- intersection(?Set1, ?Set2, ?Set3) : Το Set3 είναι τομή των Set1 και Set2.
- union(?Set1, ?Set2, ?Set3) : Το Set3 είναι ένωση των Set1 και Set2.
- difference (?Set1, ?Set2, ?Set3) : Το Set3 είναι διαφορά των Set1 και Set2.
- symdiff (?Set1, ?Set2, ?Set3) : Το Set3 είναι συμμετρική διαφορά των Set1 και Set2.

Περιορισμοί ανάμεσα σε σύνολα

Στα επόμενα το +Sets είναι **λίστα συνόλων**.

- all_disjoint(+Sets): Τα σύνολα πρέπει να είναι διάφορα μεταξύ τους.
- all_intersection(+Sets, ?Intersection): Η τομή των συνόλων είναι το σύνολο ?Intersection
- all_union(+Sets, ?SetUnion): Η ένωση των συνόλων είναι το σύνολο ?SetUnion

Περιορισμός weight

- weight(Set, ElementWeights, Weight):
  - Set είναι μια μεταβλητή συνόλων από [\]..[1..length(ElementWeights)],
  - To ElementWeights ένα array (term) με βάρη, και
  - Weight μια integer constraint μεταβλητή που εκφράζει το συνολικό άθροισμα των βαρών του Set.

Αναζήτηση σε σύνολα ακεραίων

- insetdomain(S,?CardSel, ?ElemSel, ?Order): Δίνει στη μεταβλητή συνόλου S, μια τιμή από το πεδίο της.
- Οι τρεις μεταβλητές είναι options
  - **CardSel**: απαρίθμηση με βάση τον πληθικό αριθμό του set.
    - any, increasing, decreasing
  - **ElemSel**: η σειρά με την δοκιμάζονται τα στοιχεία για το σύνολο
    - small_first (default), big_first, random, heavy_first(Weights), light_first(Weights)
  - **Order**: αν το στοιχείο που θα επιλέγει σαν πιθανό στοιχείο του σύνολου, δοκιμάζεται αν πρώτα μπαίνει στο σύνολο ή αποκλείεται από αυτό.
    - in_notin (default), notin_in, sbds, gap_sbds_in_notin, gap_sbds_notin_in, gap_sbdd_in_notin, gap_sbdd_notin_in

Παράδειγμα

- Έστω οι αριθμοί
  - [2, 4, 5, 11, 14, 17, 18, 21, …, 55, 67, 89, 98]
- οι οποίοι πρέπει να χωριστούν σε N σύνολα, καθένα από τα οποία
  - το άθροισμα των στοιχείων του κάθε συνόλου είναι μεγαλύτερο του 40.
  - έχει τουλάχιστον δύο στοιχεία
  - Όλοι οι αριθμοί πρέπει να μπουν σε κάποιο σύνολο.

Μοντελοποίηση

- Μεταβλητές συνόλων
- **disjoint** (διαφορετικά στοιχεία)
- **weight** (υπολογισμός αθροισμάτων)
- 2 τουλάχιστον στοιχεία (cardinality)
- Όλα τα στοιχεία θα πρέπει να ανατεθούν σε ένα σύνολο (cardinality).

Ορισμός Μεταβλητών

- Ορίζουμε N μεταβλητές (set variables)
  - Το domain της κάθε μεταβλητής είναι [] .. [1,2,3,..., N]
  - Τα σύνολα είναι διαφορετικά μεταξύ τους.

```
%%% Problem Data
numbers([2, 4, 5, 11, 14, 17, 18, 21, 22, 29, 34, 45, 55, 67, 89, 98]).

solve(N,Groups):-
    numbers(Nums),
    length(Nums,AllNums),
    length(Groups,N),

    intsets(Groups,N,1,AllNums),
    all_disjoint(Groups),
    ...
```

Μοντελοποίηση αθροίσματος στοιχείων πίνακα.

- Περιορισμός weight
  - Μόνη λύση για να πάρουμε το άθροισμα των στοιχείων του συνόλου.

```
    ...
    Array =..[a|Nums],
    sums(Groups,Array,40),
    ...

%%% State the sum constraint
sums([],_,_).
sums([G|Groups],Array,S):-
    weight(G,Array,W),
    W #> S,
    sums(Groups,Array,S).
```

Άλλοι περιορισμοί

- Κάθε σύνολο πρέπει να έχει τουλάχιστον δύο στοιχεία

```
    ...
    numbers(Nums),
    length(Nums,AllNums),
    ...
    cardinality_cons(Groups,Card),
    Card #= AllNums,
    ...

%%% And the cardiality.
cardinality_cons([],0).
cardinality_cons([G|Groups],Card):-
    #(G,C), C #> 2,
    cardinality_cons(Groups,RestC),
    Card #= C + RestC.
```

Ανάθεση τιμών

- Ανάθεση τιμών στα σύνολα

```
 ...
  labelSets(Groups),
  pprint(Groups,Nums).

%%% label all setsusing indomainset
labelSets([]).
labelSets([G|Groups]):-
  insetdomain(G,_,_,_),
  labelSets(Groups).
```

Print nicely

```
%%% Prints nicely the sets on screen.
pprint([],_).
pprint([G|_Groups],Nums):-
    write(G), write("::"),
    member(X,G),
    nth1(X,Nums,N),
    write(N),write(" "),
    fail.

pprint([_|Groups],Nums):-
    nl, pprint(Groups,Nums).

%%% The nth element of a list
nth1(1,[X|_],X).
 nth1(N,[_|Rest],X):-
    N > 1,
    NN is N - 1,
    nth1(NN,Rest,X).
```

Παράδειγμα: Ελαχιστοποίηση Χρόνου

- Ένα έργο πληροφορικής αποτελείται από **10 εργασίες (tasks)**, κάθε μια από τις οποίες έχει **μια καθορισμένη διάρκεια**.
- Υπάρχουν **περιορισμοί διάταξης**, πχ. η εργασία 1 πρέπει να εκτελεστεί πριν από τις εργασίες 2 και 3, η 7 μετά την 5, κοκ. Δεν υπάρχουν περιορισμοί ανάμεσα σε όλες τις εργασίες (μερική διάταξη).
- **Διαθέσιμες είναι 4 ομάδες προγραμματιστών,** όμως κάθε ομάδα <u>είναι ικανή</u> να υλοποιήσει ένα υποσύνολο από τις διαθέσιμες εργασίες.
- *Ποια είναι η ανάθεση εργασιών στις ομάδες ώστε να ελάχιστη η διάρκεια του συνολικού έργου;*

Μοντελοποίηση

- **Διαθέσιμες είναι 4 ομάδες προγραμματιστών**, όμως κάθε ομάδα είναι ικανή να υλοποιήσει ένα υποσύνολο από τις διαθέσιμες εργασίες.
- Έννοια των συνόλων σαν τιμή σε μεταβλητή περιορισμών.

Μοντελοποίηση

- Κάθε ομάδα μπορεί να εκτελέσει υποσύνολο των διαθέσιμων εργασιών

```
%%% Possible assignment to teams.
T1 :: [] .. [4,1,3,5,6,8],
T2 :: [] .. [6,3,5,2,10,1],
T3 :: [] .. [8,4,5,7,9,10],
T4 :: [] .. [3,7,8,9,2,5],
```

- Κάθε εργασία θα εκτελεστεί από μια μόνο ομάδα:
  - all_disjoint([T1,T2,T3,T4]),

Τελικός Κώδικας

```
solve(T1,T2,T3,T4):-
    Starts = [St1,St2,St3,St4,St5,St6,St7,St8,St9,St10],
    Durs = [5,4,7,1,9,1,1,4,3,5],
    Ends = [End1,End2,End3,End4,End5,End6,End7,End8,End9,End10],
    Starts #:: 0..1000, Ends #:: 0..1000,
    End1 #= 5 + St1,  End2 #= 4 + St2, End3 #= 7 + St3, End4 #= 1 + St4,
    End5 #= 9 + St5, End6 #= 1 + St6, End7 #= 1 + St7, End8 #= 4 + St8,
    End9 #= 3 + St9, End10 #= 5 + St10,
    End1 #< St2, End1 #< St3, End3 #< St8,  End5 #< St7,
    End4 #< St9,End9 #< St10,  St1 #= 0, %% First Job,
    ic:maxlist(Ends,MakeSpan),
```

Τελικός Κώδικας

```
%%% Possible assignment to teams.
 T1 :: [] .. [4,1,3,5,6,8],  T2 :: [] .. [6,3,5,2,10,1],
 T3 :: [] .. [8,4,5,7,9,10], T4 :: [] .. [3,7,8,9,2,5],
 all_disjoint([T1,T2,T3,T4]),
 bb_min((
 insetdomain(T1,_,_,_), insetdomain(T2,_,_,_),
 insetdomain(T3,_,_,_), insetdomain(T4,_,_,_),
 built_lists(T1,Starts,Durs,S1,D1), safe_disjunctive(S1,D1),
 built_lists(T2,Starts,Durs,S2,D2), safe_disjunctive(S2,D2),
 built_lists(T3,Starts,Durs,S3,D3), safe_disjunctive(S3,D3),
 built_lists(T4,Starts,Durs,S4,D4), safe_disjunctive(S4,D4),
 labeling(Starts)),MakeSpan,_),
 write("The minimum makespan is "),write(MakeSpan),nl,
 display_results(1,[T1,T2,T3,T4],Starts,Durs).
```

Άλλοι περιορισμοί

- Έστω ότι θέλω κάθε ομάδα να πάρει μια τουλάχιστον εργασία:
  - #(T1,C1), C1 #>=1,
  - #(T2,C2), C2 #>=1,
  - #(T3,C3), C3 #>=1,
  - #(T4,C4), C4 #>=1,
- … και η ομάδα 1 διπλάσιες δουλειές της ομάδας 4, ενω οι 2 και 3 ίδιο αριθμό.
  - C1#= 2 * C4, C3#=C2,