Systems and Solving Techniques for Knowledge Representation

– Aggregates & Weak constraints –

Marco Maratea
University of Genoa, Italy

066 011 Double degree programme Computational Logic
066 931 Computational Intelligence
066 937 Software Engineering & Internet Computing
Institute of Information Systems

ASP Basics

ASP:

Datalog $\leftarrow$ done!

+ Default negation $\leftarrow$ done!
+ Disjunction $\leftarrow$ done!
+ Integrity Constraints $\leftarrow$ done!
+ Weak Constraints
+ Aggregate atoms

Weak Constraints

Weak Constraints:

- express desiderata as soft constraints in CSP
- i.e., constraints which should possibly be satisfied

Syntax (Ground, Simplified): $:\sim b_1, \ldots, b_n.$

Intuitive meaning: "satisfy $:\sim b_1, \ldots, b_n$ if possible"

Weak Constraints

Weak Constraints:

- express desiderata as soft constraints in CSP
- i.e., constraints which should possibly be satisfied

Syntax (Ground, Simplified): $:\sim b_1, \ldots, b_n.$

Intuitive meaning: "satisfy $:\sim b_1, \ldots, b_n$ if possible"

Weak Constraints

Weak Constraints:

- express desiderata as soft constraints in CSP
- i.e., constraints which should possibly be satisfied

Syntax (Ground, Simplified): $:\sim b_1, \ldots, b_n.\ [w@p]$

Intuitive meaning: "satisfy $:\sim b_1, \ldots, b_n$ if possible"

Weigth and Priority: ($[w@p]$)

- higher weights/priorities $\Rightarrow$ higher importance
- "$@p$" can be omitted

Weak Constraints Example

Example (Exams Scheduling)

Problem: Assign course exams to 3 time slots avoiding overlapping of exams of courses with common students.

Strict Solution:

$assign(X, s1) \mid assign(X, s2) \mid assign(X, s3) \text{ :- } course(X).$

$\text{:- } assign(X, S), assign(Y, S), commonStudents(X, Y, N), N > 0.$

Approximate Solution:

$assign(X, s1) \mid assign(X, s2) \mid assign(X, s3) \text{ :- } course(X).$

% If overlapping is unavoidable, then reduce it “As Much As Possible”

$:\sim assign(X, S), assign(Y, S), commonStudents(X, Y, N), N > 0. \ [N@0]$

NB: Scenarios (models) minimizing the total number of “lost" exams are preferred.

Semantics of Weak Constraints

Rules(*P*): set of the rules (including facts and strong constraints) of *P*.

WC(*P*): weak constraints of *P*.

Semantics:

- **Without Priorities:**

  • Answer sets of Rules(*P*) minimizing the sum of the weights of the violated constraints in WC(*P*)

- **With Priorities:**

  • minimize the sum of the weights of the violated constraints in the highest priority level;

  • then minimize the sum of the weights of the violated constraints in the next lower level, etc.

ASP Basics

ASP:

Datalog $\leftarrow$ done!

+ Default negation $\leftarrow$ done!
+ Disjunction $\leftarrow$ done!
+ Integrity Constraints $\leftarrow$ done!
+ Weak Constraints $\leftarrow$ done!
+ Aggregate atoms

Aggregate Atom

$$L_g <_1 f\{S\} <_2 U_g$$

$$5 < \#count\{EmpId : emp(EmpId, male, Skill, Salary)\} \leq 10$$

The atom is true if the number of male employees is greater than 5 and does not exceed 10.

Formal semantics: extension of the notion of answer set.

Aggregate Functions

Example (Team Building)

% An employee is either included in the team or not
$inTeam(I) \mid outTeam(I) \text{ :- } emp(I, Sx, Sk, Sa).$

% The team consists of a certain number of employees
$\text{:- } nEmp(N), \text{not } \#count\{I : inTeam(I)\} = N.$

% At least a given number of different skills must be present in the team
$\text{:- } nSkill(M), \text{not } \#count\{Sk : emp(I, Sx, Sk, Sa), inTeam(I)\} \leq M.$

% The sum of the salaries of the employees working in the team must not exceed the given budget
$\text{:- } budget(B), \text{not } \#sum\{Sa, I : emp(I, Sx, Sk, Sa), inTeam(I)\} \leq B.$

% The salary of each individual employee is within a specified limit
$\text{:- } maxSal(M), \text{not } \#max\{Sa : emp(I, Sx, Sk, Sa), inTeam(I)\} \leq M.$

Aggregate Semantics

Reduct: The reduct **FLP** (Faber, Leone and Pfeifer) is employed.

Answer Set: An *answer set* of a program P is a set $X \subseteq B_P$ such that X is a minimal model of P^X.

- Equivalent to Gelfond & Lifschitz transformation on aggregate-free programs
- More general
- Can be used for *recursive aggregates*, *Ex-programs*, etc.

For more details on syntax and semantics ...

The ASP-Core-2 input language format can be found at:

`https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.03b.pdf`

Thanks to Francesco Ricca for a preliminary version of these slides