Aalto University
School of Science

CS-E4540 Answer Set Programming

Introduction

Aalto University
School of Science
Department of Computer Science

Autumn 2016

Introduction

1. DECLARATIVE PROBLEM SOLVING

- Declarative programming languages allow the specification of what is to be computed rather than how computation takes place.
- PROLOG (PROgramming in LOGic) is a prototypical language that was developed for declarative programming.

```
Nat(0). Nat(s(X)) :- Nat(X).
```

- Programming in a procedural language such as Pascal, C, or Java is much about controlling the execution order of commands.

```
unsigned int f(unsigned int x) {
  if(x == 0 || x==1)
    return 1;
  else return x*f(x-1);
}
```

Conceptual Model

A problem is solved using a declarative programming language by

1. modeling the problem domain using the language,
2. performing actual computation steps to produce output, and
3. extracting a solution for the problem from the output.

$$\begin{tikzcd} \boxed{\text{Problem}} \arrow[d] & \boxed{\text{Solution}} \\ \boxed{\text{Model}} \arrow[r] & \boxed{\text{Output}} \arrow[u] \end{tikzcd}$$

Compilers and/or interpreters can be used to execute the model.

Basic Requirements

Any declarative language should

- have a clear declarative semantics,
- enable concise formalization of a variety of problem domains,
- lend itself to modular program development, and
- provide sufficient performance and scalability.

Remark

The last two requirements may endanger the declarative nature of programming (cf. the use of cuts "!" in PROLOG), i.e.,
a form of control becomes necessary for efficiency reasons.

2. ANSWER SET PROGRAMMING

Answer set programming (ASP) is a paradigm for declarative programming that effectively emerged in the late nineties.

- A rule-based language is used for problem encodings.
- Every program P, i.e., a set of rules, has a clearly defined semantics (the set of answer sets associated with P).
- The order of rules and the order of individual conditions in rules is irrelevant which gives a declarative nature for answer sets.
- Dedicated search engines—answer set solvers—are used to compute an answer set or several answer sets for a program.

Revising the Conceptual Model for ASP

A problem is encoded so that the answer sets of the respective program and the solutions of the problem are in a tight correspondence.

$$
\begin{tikzcd}
\boxed{\text{Problem}} \arrow[d] & \boxed{\text{Solution}} \\
\boxed{\text{Set of rules}} \arrow[r] & \boxed{\text{Answer set}} \arrow[u]
\end{tikzcd}
$$

Current answer set solvers expect ground programs as their input which implies a preprocessing step in order to remove variables.

Example: Graph Coloring

```
edge(a,b). edge(b,c). edge(c,a).          % Edges

node(X) :- edge(X,Y).                     % Extract nodes
node(Y) :- edge(X,Y).

r(X) :- not g(X), not b(X), node(X).      % Red
g(X) :- not b(X), not r(X), node(X).      % Green
b(X) :- not r(X), not g(X), node(X).      % Blue

:- r(X), r(Y), edge(X,Y).                 % Constraints
:- g(X), g(Y), edge(X,Y).
:- b(X), b(Y), edge(X,Y).

#show r/1.
#show g/1.
#show b/1.
```

Example: Running the Solver

The program for 3-coloring graphs is used as follows:

```
$ gringo color.lp | clasp
clasp version 3.2.0
Reading from stdin
Solving...
Answer: 1
r(a) g(b) b(c)
SATISFIABLE

Models        : 1+
Calls         : 1
Time          : 0.000s (Solving: 0.00s 1st Model: 0.00s ...)
CPU Time      : 0.000s
```

Roots of ASP

- Knowledge representation and reasoning
- Databases (SQL)
- Deductive databases
- Logic programming (PROLOG)
  – SLD-Resolution
  – Negation as failure to prove
  – Clark's completion and supported models
- Constraint programming

☞ $\boxed{\text{ASP} = \text{KR} + \text{DB} + \text{Search}}$

Example: SuDoku Puzzle

```
number(1..9).
border(1). border(4). border(7).

region(X,Y) :- border(X), border(Y).

1 { value(X,Y,N): number(X), number(Y),
    X1<=X, X<=X1+2, Y1<=Y, Y<=Y1+2 } 1
  :- number(N), region(X1,Y1).

:- 2 { value(X,Y,N): number(N) }, number(X), number(Y).
:- 2 { value(X,Y,N): number(Y) }, number(N), number(X).
:- 2 { value(X,Y,N): number(X) }, number(N), number(Y).
```

Example: Running the SuDoku Program

Royle's instance with 16 clues is solved in a fraction of a second:

```
$ gringo sudoku.lp royle.lp | clasp 1
clasp version 3.2.0
Reading from stdin
Solving...
Answer: 1
value(1,3,2) value(1,9,1) ... value(7,9,4) value(8,9,2)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.012s (Solving: 0.00s 1st Model: 0.00s ...)
CPU Time     : 0.000s
```

Example

The corresponding solution can be extracted from the answer set and then visualized as a solved SuDoku puzzle:

1	9	3	8	6	7	4	2	**5**
4	6	8	5	**3**	2	9	1	7
7	5	**2**	1	**4**	9	6	8	3
6	2	1	4	7	3	5	9	8
5	**3**	**4**	9	1	8	**7**	6	2
9	8	7	**2**	5	**6**	3	4	**1**
2	1	6	3	9	**5**	8	7	4
8	**7**	5	6	2	4	1	**3**	9
3	4	9	7	8	**1**	2	5	6

Example

Actually, there are 2 solutions for this 16 clue puzzle. The other is obtained by exchanging the occurrences of 8 and 9:

1	8	3	9	6	7	4	2	**5**
4	6	9	5	**3**	2	8	1	7
7	5	**2**	1	**4**	8	6	9	3
6	2	1	4	7	3	5	8	9
5	**3**	**4**	8	1	9	**7**	6	2
8	9	7	**2**	5	**6**	3	4	**1**
2	1	6	3	8	**5**	9	7	4
9	**7**	5	6	2	4	1	**3**	8
3	4	8	7	9	**1**	2	5	6

Applications of ASP

- Argumentation
- Code (super)optimization
- Configuration
- Cryptanalysis
- Database integration
- Decision support
- Diagnosis
- Games
- Music composition
- Phylogenetics
- Planning
- Semantic web
- Testing and verification
- Timetabling

(Abridged from `http://www.cs.uni-potsdam.de/~torsten/asp/`)

Application: Learning Markov Networks

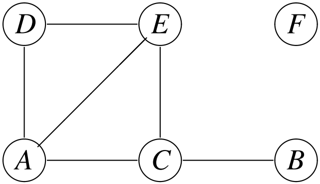
Example

Heart disease dataset [Edwards and Havránek, 1985]:

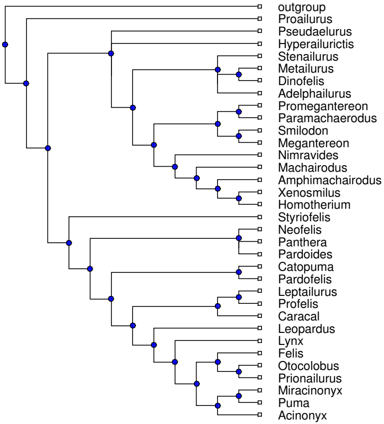
A	Smoking
B	Strenuous mental work
C	Strenuous physical work
D	Systolic blood pressure > 140
E	Ratio of beta and alpha lipoproteins > 3
F	Family anamnesis of coronary heart disease

Resulting Markov Network

A	B	C	D	E	F
1	1	0	1	0	0
0	0	0	1	0	0
0	1	1	0	1	1
0	1	1	0	0	1
0	0	0	1	0	0
0	0	0	1	0	1
0	1	0	1	0	0
1	0	1	0	1	0
0	0	0	1	1	0
1	0	0	0	1	1
0	1	1	1	0	1
0	1	1	0	0	1
0	0	1	1	0	0
0	0	0	0	1	0
⋮	⋮	⋮	⋮	⋮	⋮



Application: Supertree Construction



Factors Behind the Success of ASP

- The performance of computers has increased remarkably.
- Implementation techniques have advanced rapidly.
- Many efficient solvers are publicly available.

  Download state-of-the-art tools from
  `http://potassco.sourceforge.net/`

  — the `gringo` grounder (4.5.4) and
  — the `clasp` solver (3.2.0).

- Rule-based languages are highly expressive—enabling concise encodings for a wide variety of problems.
- ASP languages lend themselves to fast prototyping with little programming effort.

3. SOME PREREQUISITES

- Propositional languages
- Interpretations and models
- Logical entailment
- First-order languages
- Structures
- Herbrand bases
- Herbrand structures and models
- Relational operations

Propositional Languages

- Any set of atomic sentences $\mathcal{P} \neq \emptyset$, or atoms for short, induces a propositional language $\mathcal{L}$— the set of well-formed sentences.
- Sentences are built using the atoms of $\mathcal{P}$ and propositional connectives $\neg$ (negation), $\land$ (conjunction), $\lor$ (disjunction), $\rightarrow$ (implication), and $\leftrightarrow$ (equivalence).
  1. Atomic sentences are sentences.
  2. If α and β are sentences, then expressions of the forms $(\neg\alpha)$, $(\alpha \lor \beta)$, $(\alpha \land \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \leftrightarrow \beta)$ are also sentences.
- Propositional theories T are defined as subsets of $\mathcal{L}$.

Example

The theory $T = \{r \lor g \lor b,\ \neg r \lor \neg g,\ \neg g \lor \neg b,\ \neg b \lor \neg r\}$ describes the 3-coloring of a single node in a graph.

Interpretations and Models

- An interpretation I for $\mathcal{L}$ is defined as any subset of $\mathcal{P}$:
  1. atoms in I are considered to be true and
  2. atoms in $\mathcal{P} \setminus I$ are false.
- If $\mathcal{P}$ is finite, there are $|\mathbf{2}^{\mathcal{P}}| = 2^{|\mathcal{P}|}$ different interpretations, each of which represents a unique state of the world described in $\mathcal{L}$.
- The satisfaction $I \models \alpha$ of a sentence $\alpha \in \mathcal{L}$ in an interpretation I is defined in the standard way.
- An interpretation I is a model of a theory T iff $I \models T$, i.e., $I \models \alpha$ holds for every $\alpha \in T$.

Example

The theory $T = \{r \vee g \vee b,\ \neg r \vee \neg g,\ \neg g \vee \neg b,\ \neg b \vee \neg r\}$ based on $\mathcal{P} = \{r, g, b\}$ has models $M_1 = \{r\}$, $M_2 = \{g\}$, and $M_3 = \{b\}$.

Logical Entailment

- A sentence α is a logical consequence of a theory T, denoted $T \models \alpha$, iff $M \models \alpha$ for every model $M \models T$.
- The set of logical consequences $\mathrm{Cn}(T) = \{\alpha \in \mathcal{L} \mid T \models \alpha\}$.
- The operator $\mathrm{Cn}(\cdot)$ has the properties of a closure operator. For any T_1 and T_2,
  1. $T_1 \subseteq \mathrm{Cn}(T_1)$,
  2. $T_1 \subseteq T_2 \implies \mathrm{Cn}(T_1) \subseteq \mathrm{Cn}(T_2)$, and
  3. $\mathrm{Cn}(\mathrm{Cn}(T_1)) = \mathrm{Cn}(T_1)$.

Example

Consider the theory $T = \{a,\ a \rightarrow b,\ \neg b \vee c,\ d \rightarrow \neg c\}$ based on $\mathcal{P} = \{a,b,c,d\}$. The theory has a unique model $M = \{a,b,c\}$.
$\implies \mathrm{Cn}(T) = \{a,\ a \rightarrow b,\ b,\ \neg b \vee c,\ c,\ d \rightarrow \neg c,\ \neg d,\ c \vee d, \ldots\}$.

First-Order Languages (I)

- A first-order language $\mathcal{L}$ is based on mutually disjoint sets of
  – constant symbols $\mathcal{C}$,
  – variable symbols $\mathcal{V}$,
  – function symbols $\mathcal{F}$, and
  – relation symbols $\mathcal{R}$.
- A term is either
  1. a constant symbol c from $\mathcal{C}$,
  2. a variable symbol v from $\mathcal{V}$, or
  3. an expression of the form $f(t_1,\ldots,t_n)$ where f is a function symbol of arity $n > 0$ from $\mathcal{F}$ and $t_1,\ldots,t_n$ are terms.

Remark

Constants represent function symbols of arity 0.

First-Order Languages (II)

- An atomic formula is an expression of the form
  1. R for each relation symbol of arity 0 from $\mathcal{R}$,
  2. $t_1 = t_2$ where t_1 and t_2 are terms, or
  3. $R(t_1, \ldots, t_n)$ where R is a relation symbol of arity $n > 0$ from $\mathcal{R}$ and $t_1, \ldots, t_n$ are terms.
- Atomic formulas are formulas.
- If α and β are formulas and v is a variable from $\mathcal{V}$, then

  $$(\neg\alpha),\ (\alpha \vee \beta),\ (\alpha \wedge \beta),\ (\alpha \rightarrow \beta),\ (\alpha \leftrightarrow \beta),\ (\forall v \alpha),\ \text{and}\ (\exists v \alpha)$$

  are also formulas.
- A sentence is a formula having no free occurrences of variables.

Structures (I)

- An interpretation for a first-order language $\mathcal{L}$ is a structure S based on a universe U which is any non-empty set and
  1. each $c \in \mathcal{C}$ is mapped to an element $c^S \in U$,
  2. each $v \in \mathcal{V}$ is mapped to an element $v^S \in U$,
  3. each $f \in \mathcal{F}$ is mapped to a function $f^S : U^n \to U$ where n is the arity of f, and
  4. each $R \in \mathcal{R}$ with an arity n is mapped to a relation $R^S \subseteq U^n$.
- Given a structure S, each term t is mapped to an element $t^S \in U$.

Example

Given a constant symbol 0 and a unary (of arity 1) function symbol s we may define a structure S based on $U = \{0, 1, 2, \ldots\}$ by setting $0^S = 0$ and $s^S : x \mapsto x + 1$. Thus $(s(s(s(0))))^S = 3$.

Structures (II)

- Atomic formulas R, $t_1 = t_2$, and $R(t_1, \ldots, t_n)$ are satisfied by S iff $\langle\rangle \in R^S$, $(t_1)^S = (t_2)^S$, and $\langle (t_1)^S, \ldots, (t_n)^S \rangle \in R^S$, respectively.
- The satisfaction of a first order formula/sentence α in a structure is defined in the standard way.
- Structures that are models of sentences ($S \models \alpha$) and theories ($S \models T$) are distinguished in analogy to propositional logic.
- The definition of $T \models \alpha$, i.e., whether a sentence α is a logical consequence of a theory T, remains unchanged.

Example

For $T = \{E(0),\ \forall x(E(x) \rightarrow O(s(x))),\ \forall x(O(x) \rightarrow E(s(x)))\}$ formalizing even natural numbers: $T \models E(s(s(0)))$ but $T \not\models \neg E(s(0))$.

Herbrand Bases

- A ground term is a term having no occurrences of variables.
- Given the sets $\mathcal{C}$ and $\mathcal{F}$ (see above), the Herbrand universe is the set of ground terms constructible using the symbols of $\mathcal{C}$ and $\mathcal{F}$.
- Given the set $\mathcal{R}$, the Herbrand base consists of atomic sentences $R(t_1,\ldots,t_n)$ where $R \in \mathcal{R}$ is of arity n and each t_i is a ground term.
- A Herbrand instance of an atomic formula $R(t_1,...,t_n)$ is obtained by substituting ground terms for variables occurring in $t_1,\ldots,t_n$.
- We may also define the Herbrand base $\mathrm{Hb}(T)$ of a theory T by inspecting which constant/function symbols occur in T.

Example

For the previous theory T formalizing even natural numbers, we have $\mathrm{Hb}(T) = \{E(0), O(0), E(s(0)), O(s(0)), \ldots\}$.

Herbrand Structures and Models

- A Herbrand structure S is based on a fixed interpretation of constant and function symbols over the Herbrand universe:
  1. Each $c \in \mathcal{C}$ is mapped to $c^S = c$.
  2. Each $f \in \mathcal{F}$ is mapped to $f^S : \langle t_1, \ldots, t_n \rangle \mapsto f(t_1, ..., t_n)$.

  $\Longrightarrow$ Only interpretations of variables and relations can vary!
- Any Herbrand structure S can be viewed as a propositional interpretation $I = \{R(t_1, \ldots, t_n) \in \mathrm{Hb}(T) \mid S \models R(t_1, ..., t_n)\}$.
- A Herbrand model M of a theory T is a Herbrand structure that satisfies all sentences of T.

Example

For the theory T formalizing even natural numbers, we have a Herbrand model $M = \{E(0), O(s(0)), E(s(s(0))), O(s(s(s(0)))), \ldots\}$.

Relational Operations

Assume that R_1 and R_2 are binary relations (of arity 2) over a fixed universe U, i.e., $R_1 \subseteq U^2$ and $R_2 \subseteq U^2$.

1. The union of R_1 and R_2 is
   $R_1 \cup R_2 = \{\langle x,y\rangle \in U^2 \mid \langle x,y\rangle \in R_1$ or $\langle x,y\rangle \in R_2\}$.
2. The intersection of R_1 and R_2 is
   $R_1 \cap R_2 = \{\langle x,y\rangle \in U^2 \mid \langle x,y\rangle \in R_1$ and $\langle x,y\rangle \in R_2\}$.
3. The projections of R_1 w.r.t. the first/second arguments are
   $P_1 = \{x \in U \mid \langle x,y\rangle \in R_1\}$ and $P_2 = \{y \in U \mid \langle x,y\rangle \in R_1\}$.
4. The composition of R_1 of R_2 is
   $R_1 \circ R_2 = \{\langle x,y\rangle \in U^2 \mid \langle x,z\rangle \in R_1$ and $\langle z,y\rangle \in R_2\}$.

OBJECTIVES

- You have the necessary premises for the course, i.e., you are familiar with the syntax and semantics of classical logic.
- You know the main characteristics of declarative programming languages and understand the difference w.r.t. procedural ones.
- You understand the conceptual model of answer set programming.
- You are able to list the basic steps which are required to to apply ASP in declarative problem solving.

TIME TO PONDER

Definition

The set of classical models associated with a propositional theory T is $\mathrm{CM}(T) = \{M \subseteq \mathrm{Hb}(T) \mid M \models T\}$.

Problem

Let T_1 and T_2 be arbitrary propositional theories which may be based on different Herbrand bases $\mathrm{Hb}(T_1)$ and $\mathrm{Hb}(T_2)$.
Which one of the following equations is correct in general?

1. $\mathrm{CM}(T_1 \cup T_2) = \mathrm{CM}(T_1) \cap \mathrm{CM}(T_2)$.
2. $\mathrm{CM}(T_1 \cup T_2) = \{M_1 \cup M_2 \mid M_1 \in \mathrm{CM}(T_1)$ *and* $M_2 \in \mathrm{CM}(T_2)\}$.
3. $\mathrm{CM}(T_1 \cup T_2) = \{M_1 \cup M_2 \mid$
   $M_1 \in \mathrm{CM}(T_1),\ M_2 \in \mathrm{CM}(T_2),$ *and* $M_1 \cap C = M_2 \cap C\}$
   where $C = \mathrm{Hb}(T_1) \cap \mathrm{Hb}(T_2)$ *gives atoms common to* T_1 *and* T_2.