Lecture 1: Introduction

Outline
1. Declarative problem solving
2. Answer set programming
3. Some prerequisites

1. DECLARATIVE PROBLEM SOLVING

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

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

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 necessary for efficiency reasons.

Conceptual Model

A problem is solved using a declarative programming language by
1. *modelling* 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.

```
Problem   Solution
  ↓       ↑
Model    → Output
```

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.*
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.

Example: Graph Coloring

define edge(a,b), edge(b,c), edge(c,a).
% Edges
define node(X) :- edge(X,Y).
% Extract nodes
define 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).

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

$ lparse color.lp > color.sm$
$ lplist color.sm | less$
$ smodels 1 < color.sm$

Answer: 1
Stable Model: r(a) g(c) b(b) edge(c,a) edge(b,c) edge(a,b) \
ode(c) node(b) node(a)
True
Duration: 0.004
Number of choice points: 2
Number of wrong choices: 0
Number of atoms: 16
Number of rules: 24
Number of picked atoms: 22
Number of forced atoms: 0
Number of truth assignments: 77
Size of searchspace (removed): 9 (0)

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.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set of rules</td>
<td>Answer set</td>
</tr>
</tbody>
</table>

Current answer set solvers expect ground programs as their input which implies a preprocessing step in order to remove variables.
Example. Royle’s SuDoku puzzle with 16 clues gets solved in 52 ms.

```bash
$ lparse sudoku.lp royle.lp | smodels 1smodels version 2.32. Reading...done
Answer: 1
Stable Model: value(8,8,1) value(4,7,1) ... value(3,1,9)
True
Duration: 0.052
Number of choice points: 1
Number of wrong choices: 0
Number of rules: 928
Number of picked atoms: 321
Number of forced atoms: 51
Number of truth assignments: 8017
Size of searchspace (removed): 36 (0)
```

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:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Factors Behind the Success of ASP

- The performance of computers has increased remarkably.
- Implementation techniques have advanced rapidly.
- Many efficient solvers are publicly available: smodels, clasp, cmodels, dlv, ...
- 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.

Applications of ASP

- Product configuration
- Combinatorial problems
  - Graph problems, combinatorial auctions, ...
- AI Planning
  - Contingency plans for the NASA space shuttle
- Verification and analysis
  - Communication protocols, security protocols, ...
- Information and data integration
  - Semantic web

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 $\alpha$ and $\beta$ 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 $|\mathcal{P}| = 2^{|\mathcal{P}|}$ 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 $T = \{r \lor g \lor b, \neg r \lor \neg g, \neg g \lor \neg b, \neg b \lor \neg r\}$ based on $\mathcal{P} = \{r, g, b\}$ has models $M_1 = \{r\}$, $M_2 = \{g\}$, and $M_3 = \{b\}$.

**Logical Entailment**

- A sentence $\alpha$ 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 $\text{Cn}(T) = \{\alpha \in \mathcal{L} \mid T \models \alpha\}$.
- The operator $\text{Cn}(\cdot)$ has the properties of a closure operator.
  1. $T_1 \subseteq \text{Cn}(T_1)$,
  2. $T_1 \subseteq T_2 \implies \text{Cn}(T_1) \subseteq \text{Cn}(T_2)$, and
  3. $\text{Cn}(\text{Cn}(T_1)) = \text{Cn}(T_1)$.

**Example.** Consider the theory $T = \{a, a \rightarrow b, \neg b \lor c, d \rightarrow \neg c\}$ based on $\mathcal{P} = \{a, b, c, d\}$. The theory has a unique model $M = \{a, b, c\}$.

$\iff \text{Cn}(T) = \{a, a \rightarrow b, b \lor c, d \rightarrow \neg c, \neg d, c \lor d, \ldots\}$

**First-Order Languages (1)**

- 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 $\alpha$ and $\beta$ are formulas and $v$ is a variable from $\mathcal{V}$, then expressions of the forms
  $(\neg \alpha)$, $(\alpha \lor \beta)$, $(\alpha \land \beta)$, $(\alpha \rightarrow \beta)$, $(\forall \alpha)$, and $(\exists \alpha)$ are also formulas.
- A sentence is a formula having no free occurrences of variables.

Structures (II)

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

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

Structures (I)

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

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

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 $R(t_1, \ldots, t_n)$ is obtained by substituting ground terms for variables occurring in $t_1, \ldots, t_n$.
- We may also define the Herbrand base $Hb(T)$ 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 $Hb(T) = \{ E(0), O(0), E(S(0)), O(S(0)), \ldots \}$. 

© 2007 TKK / TCS
**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 $c \in C$ is mapped to $c^S = c$.
  2. Each $f \in F$ is mapped to $f^S : (t_1, \ldots, t_n) \mapsto f(t_1, \ldots, t_n)$.

$\implies$ Only interpretations of variables and predicates can vary!

- Any Herbrand structure $S$ can be viewed as a propositional interpretation $I = \{ R(t_1, \ldots, t_n) \in Hb(T) \mid S \models R(t_1, \ldots, t_n) \}$.

- 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 $R_1 \cup R_2 = \{ (x, y) \in U^2 \mid (x, y) \in R_1 \text{ or } (x, y) \in R_2 \}$.

2. The *intersection* of $R_1$ and $R_2$ is $R_1 \cap R_2 = \{ (x, y) \in U^2 \mid (x, y) \in R_1 \text{ and } (x, y) \in R_2 \}$.

3. The *projections* of $R_1$ w.r.t. the first/second arguments are $P_1 = \{ x \in U \mid (x, y) \in R_1 \}$ and $P_2 = \{ y \in U \mid (x, y) \in R_1 \}$.

4. The *composition* of $R_1$ of $R_2$ is $R_1 \circ R_2 = \{ (x, y) \in U^2 \mid (x, z) \in R_1 \text{ and } (z, y) \in R_2 \}$.

**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 apply ASP in declarative problem solving.

**TIME TO PONDER**

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

**Problem.** Let $T_1$ and $T_2$ be arbitrary propositional theories. Which one of the following is correct in general?

1. $CM(T_1 \cup T_2) = CM(T_1) \cap CM(T_2)$.

2. $CM(T_1 \cup T_2) = \{ M_1 \cup M_2 \mid M_1 \in CM(T_1) \text{ and } M_2 \in CM(T_2) \}$.

3. $CM(T_1 \cup T_2) = \{ M_1 \cup M_2 \mid M_1 \in CM(T_1), M_2 \in CM(T_2), \text{ and } M_1 \cap H = M_2 \cap H \}$

where $H = Hb(T_1) \cap Hb(T_2)$. 