From Games to Logic Programs and Back to Games

Marina De Vos
In Collaboration with Jonty Needham and Dirk Vermeir

23 April 2009

Outline

Answer Set Programming Paradigm

- Fundamental concept:

  Models, not proofs, represent solutions!

- Therefore, need techniques to compute models (not to compute proofs)

  ASP Solvers

- What is model generation good for?

  Solve search problems

  - Reasoning about ontologies
  - Error diagnoses for a faulty system ;
  - Music Synthesis
  - Evolution of language
  - Optimal code sequences
  - Agent Reasoning

AnsProlog

- Clause

Definition

$a :- b_1, \ldots, b_m,$ **not** $c_1, \ldots,$ **not** c_n

- a, b_i, c_j are atoms
- a is the head of the clause, denoted $H(r)$
- $b_1, \ldots, b_m,$ **not** $c_1, \ldots,$ **not** c_n form the body of the rule, $B(r)$.
- The set of all atoms is referred to as the Herbrand base HB

Negation as Failure

- **not** c_i stands for negation as failure.
- **not** c_i is considered true if we cannot find evidence to support the truth of c_i.
- This is different from classical negation $\neg a$ where the proof a needs to exist to determine the value of $\neg a$

Example

Compare:

`innocent` : $-$**not** `evidence` with:

innocent $\leftarrow \neg$*evidence*

Answer Sets for Positive Programs

- Programs without negation as failure

Definition

The deductive closure is defined as:

1. take all rules off which the bodies are true
2. assume the heads to be true
3. continue until you reach a fixpoint

Definition

This fixpoint is the unique answer set, $lm(P)$ of the program.

An example

Example

```
a  :-  c, d
c  :-  e
f  :-  g
d
e
```

$\Rightarrow \{d,\ e, c, a\}$

Gelfond Lifschitz Reduct

Definition

Let P be a logic program and let M is a set of atoms.
The Gelfond-Lifschitz (GL) Reduct P^M is obtained as follows:

- remove each rule with some negative literal not a in the body such that $a \in M$
- remove all negative literals from all remaining rules

- The reduct P^M is a positive program
- It has the least model $lm(P^M)$

Answer Sets

Definition

Let P be a program and let M be a set of atoms ($M \subseteq HB_P$). M is an answer set of P if and only if $M = lm(P^M)$

Intuition

- M makes an assumption about what is true and what is false
- P^M derives positive facts under the assumption of **not** as by M
- If the result is M, then the assumption of M is stable and M is called an answer set.

Example

Example

$\Pi =$

$$
\begin{aligned}
a &\leftarrow b.\\
b &\leftarrow .\\
c &\leftarrow \text{not } d, a.\\
d &\leftarrow \text{not } c, a.\\
e &\leftarrow d.
\end{aligned}
$$

Example

Example

$\Pi = \{a \leftarrow b.\ b \leftarrow .\ c \leftarrow \text{not } d, a.\ d \leftarrow \text{not } c, a.\ e \leftarrow d.\}$

$X = \{a, c, e\}$

$$\begin{aligned} a &\leftarrow b. \\ b &\leftarrow . \\ c &\leftarrow \text{not } d, a. \\ d &\leftarrow \text{not } c, a. \\ e &\leftarrow d. \end{aligned}$$

$Cn(\Pi^X) = \{b, a, c\} \neq X$ X not an answer set

Example

Example

$\Pi = \{a \leftarrow b.\ b \leftarrow .\ c \leftarrow \text{not } d, a.\ d \leftarrow \text{not } c, a.\ e \leftarrow d.\}$

$X = \{a, b, d, e\}$

$$
\begin{aligned}
a &\leftarrow b. \\
b &\leftarrow . \\
c &\leftarrow \text{not } d, a. \\
d &\leftarrow \text{not } c, a. \\
e &\leftarrow d.
\end{aligned}
$$

$Cn(\Pi^X) = \{b, a, d, e\} = X$ X is an answer set

A Few Extra Constructs...

Constraints $\bot \leftarrow a, b.$

If we know a and b we don't have an answer set

Predicates $a(X, Y) \leftarrow b(X), c(Y).$

$a(1,1) \leftarrow b(1), c(1).$, $a(1,2) \leftarrow b(1), c(2).$,

$a(2,2) \leftarrow b(2), c(2).$, ...

Choice Rules $1\ \{a(1), a(2), a(3), a(4)\}\ 3 \leftarrow b.$

If we know b then we know between 1 *and* 3 *of*

$a(1), a(2), a(3), a(4)$

And functions, classical negation, variable domains, preferences, ...

Graph Colouring

Example

$$1\ \{paint(N, C) : colour(C)\}\ 1 \quad :- \quad node(N)$$
$$\bot \quad :- \quad node(N1), node(N2), N1 \neq N2, link(N1, N2), paint(N1, C), paint(N2, C), colour(C).$$

Per instance add *colour*, *node* and *link*.

Choice Logic Programming

- Variation of answer set programming
- Rules of the form: $a_1 \oplus \ldots \oplus a_n \leftarrow b_1, \ldots b_m \quad .$
- No explicit negation
- Gelfond-Lifschitz transformation
  - remove all false atoms from the head of each rule r with more than one head atom; and
  - replace all rules r where $|H_r| > 1$ (after deletion) with the following constraint:

$$\leftarrow B_r, H_r \quad .$$

- CLP programs can be mapped to normal AnsProlog

Strategic Games I

Definition

A strategic game is a tuple $\langle N, (A_i)_{i \in N}, (\succeq_i)_{i \in N} \rangle$ where

- N is a finite set of players;
- for each player $i \in N$, A_i is a nonempty set of actions that are available to her (we assume that $A_i \cap A_j = \emptyset$ whenever $i \neq j$) and,
- for each player $i \in N$, $\succeq_i$ is a preference relation on $A = \times_{j \in N} A_j$

An element $\mathbf{a} \in A$ is called a profile.
Notations: $\mathbf{a}_i$, A_{-i}, $(\mathbf{a}_{-i}, a_i)$

Strategic Games II

Definition

A Nash equilibrium of a strategic game $\langle N, (A_i)_{i \in N}, (\succeq_i)_{i \in N} \rangle$ is a profile $\mathbf{a}^*$ satisfying

$$\forall a_i \in A_i \cdot (\mathbf{a}^*_{-i}, \mathbf{a}^*_i) \succeq_i (\mathbf{a}^*_{-i}, a_i)$$

Definition

Let $\langle N, (A_i)_{i \in N}, (\succeq_i)_{i \in N} \rangle$ be a strategic game. The best response function B_i for player $i \in N$ is defined by

$$B_i(\mathbf{a}_{-i}) = \{a_i \in A_i \mid \forall a'_i \in A_i \cdot (\mathbf{a}_{-i}, a_i) \succeq_i (\mathbf{a}_{-i}, a'_i)\}$$

Strategic Games III

Example

Let us consider the Bach or Stravinsky game:

	Bach	*Stravinsky*
Bach	2, 1	0, 0
Stravinsky	0, 0	1, 2

This game has two Nash equilibria, namely:

$$(Bach_1, Bach_2) \text{ and } (Stravinsky_1, Stravinsky_2)$$

CLP'S and Games I

Definition

Let $G = \langle N, (A_i)_{i \in N}, (\geq_i)_{i \in N} \rangle$ be a strategic game. The choice logic program P_G associated with G contains the following rules:

- For each player i, P_G contains the rule $A_i \leftarrow$. This rule ensures that each player i chooses exactly one action from A_i.
- For each player i and for each profile $\mathbf{a} \in A_{-i}$, P_G contains a rule $B_i(\mathbf{a}) \leftarrow \mathbf{a}$. It models the fact that a player will select a "best response", given the other players' choices.

CLP'S and Games

Theorem

For every strategic game $G = \langle N, (A_i)_{i \in N}, (\succeq_i)_{i \in N} \rangle$ there exists a choice logic program P_G such that the set of answer sets of P_G coincides with the set of Nash equilibria of G.

Example

Example

The CLP corresponding the BoS game of example 15 equals:

$$
\begin{aligned}
b_1 \oplus s_1 &\leftarrow . \\
b_2 \oplus s_2 &\leftarrow . \\
b_1 &\leftarrow b_2 . \\
s_1 &\leftarrow s_2 . \\
b_2 &\leftarrow b_1 . \\
s_2 &\leftarrow s_1 .
\end{aligned}
$$

This program has two answer sets, namely $\{s_1, s_2\}$ and $\{b_1, b_2\}$ which correspond to the Nash equilibria of the game.

Ordered Choice Logic Programming

Definition

An Ordered Choice Logic Program, or OCLP, is a pair $\langle \mathcal{C}, \prec \rangle$ where $\mathcal{C}$ is a finite set of choice logic programs, called components, and "$\prec$" is a strict pointed partial order on $\mathcal{C}$.

- the choice rules determine the alternatives of a decision
- as before an exclusive choice has to be made
- the most preferred eligible alternative is chosen

Semantics

- Gelfond-Lifschitz reduct
- The OCLP program is reduced to a CLP by
- removing those rules with head atoms that are less preferred alternatives of a decision

Example

Consider the program with four components $C1 = \{\mathtt{a} \leftarrow \mathtt{c}\}$, $C2 = \{\mathtt{b} \leftarrow\}$, $C3 = \{\mathtt{a} \leftarrow \mathtt{d}\}$ and $C4 = \{\mathtt{a} \oplus \mathtt{b} \leftarrow; \mathtt{c} \oplus \mathtt{d} \leftarrow\}$ with $C4 \prec C3 \prec C2 \prec C1$. This program has two answer sets: $\{\mathtt{a}, \mathtt{d}\}$ and $\{\mathtt{b}, \mathtt{c}\}$.

- OCLP programs can be mapped back to standard AnsProlog

Extensive Games with Perfect Information I

Definition

A (finite) extensive game with perfect information is a tupple $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$ with the following components:

- A set N of players.
- A prefix-closed set H of finite sequences. Each element of H is called a history; each component of a history is an action taken by player. A history $a_1, \ldots, a_k$ is terminal if $\nexists a_{k+1} \cdot a_1 \ldots a_k a_{k+1} \in H$. We use Z to denote the set of terminal histories.

Extensive Games with Perfect Information II

- *A function P that assigns to each nonterminal history from $H \setminus Z$ a member of N. (P is the player function, $P(h)$ being the player making a decision after history h.).*
- *For each player $i \in N$ a preference relation $\succeq_i$ on Z (the preference relation of player i).*

A strategy for a player $i \in N$ is a function that assigns an action of $A(h)$ to each nonterminal history $h \in (H \setminus Z)$ for which $P(h) = i$.
A strategy profile s is a set containing a strategy for each player $i \in N$, i.e. $s = (s_i)_{i \in N}$.
The outcome $O(s)$ for a strategy profile s is defined as the terminal history that results when each player $i \in N$ follows the precepts of s_i. That is, $O(s)$ is the history $a_1, \ldots a_k \in Z$ such that for $0 \leq l < k$ we have $s_{P(a_1,\ldots,a_l)}(a_1, \ldots, a_l) = a_{l+1}$.

Nash Equilibria

Definition

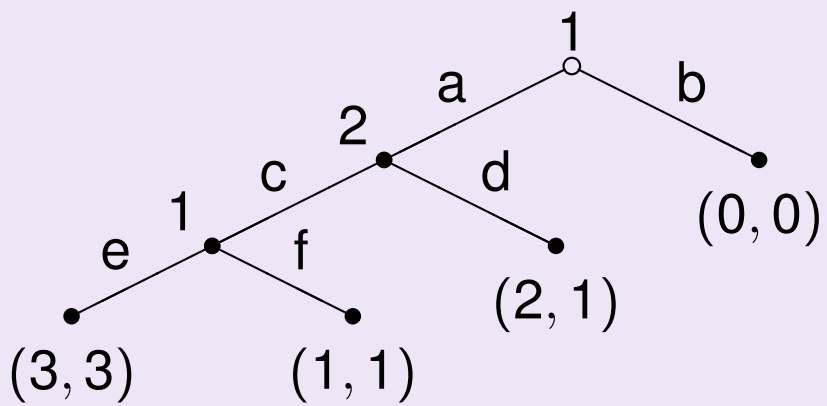
A Nash Equilibrium of an extensive game with perfect information $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$ is a strategy profile s^* such that for every player $i \in N$ we have

$$O((s^*_{-i}, s^*_i)) \succeq_i O((s^*_{-i}, s_i)) \text{ for every strategy } s_i \text{ of player } i.$$

Extensive Games with Perfect Information

Example

Consider the following extensive game:



This game has two Nash equilibria: $\{\{a, e\}, \{c\}\}$ and $\{\{a, f\}, \{d\}\}$.

Nash Equilibria and OCLP

Definition

Let $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$ be an extensive game with perfect information. The corresponding OCLP P_n can be constructed in the following way:

- $C = \{C^t\} \cup \{C_u \mid \exists i \in N, h \in Z \cdot u = U_i(h)\}$;
- $C^t \prec C_u$ for all $C_u \in C$;
- $\forall C_u, C_w \in C \cdot C_u \prec C_w$ iff $u > w$;
- $\forall h \in (H \setminus Z) \cdot (\{a \mid ha \in H\} \leftarrow) \in C^t$;
- $\forall h = h_1 a h_2 \in Z \cdot a \leftarrow B \in C_u$ with $B = \{b \in [h] \mid h = h_3 b h_4, P(h_3) \neq i\}$ and $u = U_{P(h_1)}(h)$.

Equilibria and Models

Theorem

Let $G = \langle N, H, P, (\succeq_i)_{i \in N} \rangle$ be a finite extensive game with perfect informationand let P_n be its corresponding OCLP. Then, s^* is a Nash equilibrium for G iff s^* is an answer set for P_n.

Further Extensions

- Subgame perfect equilibria for extensive games with perfect information
- Extensive game with simultaneous moves
- Mixed strategy Nash equilibria for strategic games
- Multi-agent system

Games Semantics

- Intuitive, close match with the syntax
- Very versatile mathematically
- Provides a very concrete way of building fully abstract models.
  - Applications include Linear Logic, PCF (Hyland, Ong), Idealized Algol (McCusker, Ghica)
  - Hardware design, security protocols.
- Algorithmic applications – direct from semantics
- A single-typed system

Arenas I

- Arenas: $\mathcal{A} = \langle M, \lambda, \vdash \rangle$ for a giving program Herbrand base of a program P
- $\mathcal{L} = HB_\Pi \cup \textbf{not } HB_\Pi$
- M is the set of available moves:
  - $M^p = \mathcal{L} \cup \{a? | a \in HB_\Pi\} \cup \{a \leftarrow B? | a \in HB_\Pi,\ B \subset \mathcal{L}\} \cup \{\top, \bot\} \cup \{res(i)\}$
  - $M = M^p \cup \{\circlearrowleft^m | m \in M^p\}$
- $\lambda : M \rightarrow \{O, P\} \times \{Q, A\}$.

$$\lambda(m) = \begin{cases} (O, Q) & \text{if } m = a?,\ a \in HB_\Pi \\ (P, Q) & \text{if } m = r?, r = a \leftarrow B,\ a \in HB_\Pi,\ B \subset \mathcal{L} \\ (O, A) & \text{if } m \in \{\top, \bot\} \\ (P, A) & \text{if } m \in \mathcal{L} \end{cases}$$

Arenas II

- The enabling relation $\vdash$ on $M \cup \{\star\}$ guarantees that to every answer there is a corresponding question and answers can only happen in response to the questions as follows:
  - $\star \vdash a?$
  - $r? \vdash \top$
  - $r? \vdash \bot$
  - $a? \vdash a$
  - $a? \vdash$ **not** a
  - $a? \vdash \circlearrowleft$

Rules

- Sequence of moves: $s = m_1, \ldots, m_n \in M^*$.
  - Only the latest open question may be answered.
  - Both players have full access to the history (unless in backtracking – later)
  - Play is fair.
  - O plays all moves available to him.
  - Players cannot play inconsistent moves
  - The final player to play wins.

Strategy I

- Predertimined plan for the game for one player.
- Encompasses the whole game: $\sigma : M^* \rightarrow M$
- Winning strategies:
  - A strategy σ is said to be winning when regardless of the other player's moves, the play will result in a win for the player.
  - P can play any $m \in \{a \leftarrow B? | a \in HB, B \subset \mathcal{L}\}$, but only certain moves are useful.
  - These useful moves are characterized by *AnsProlog* programs with winning strategies.

Strategy II

- $\sigma := [\![\Pi]\!]$ is the strategy derived from the program Π such that:
- Let $s \in M*$ and m_n the last element
  - If O plays $a?$ and $a \notin \{l = head(r) | r \in \Pi\}$ then P plays **not** a
  - If O plays $a?$ and $a \in \{l = head(r) | r \in \Pi\}$ then P plays $r = a \leftarrow B \in \Pi$
  - $\sigma(s) = a$ if $m_n = \top$ and the pending question is $a?$
  - $\sigma(s) =$ **not** a if $m_n = \bot$ and the pending question is $a?$ and there is no further move available of the form $a \leftarrow B?$. Otherwise, $\sigma(s) = r?$ with $r \in \{r | a = head(r) \wedge r \in \Pi\}$.
  - $\sigma(s) = a$ if a is proved by the application of some rule, even though the pending rule may not be applied.
  - Structural move: The backtracking move, $\circlearrowleft$ can be played immediately upon P being forced to play $\bot$. Instead of playing $\bot$, P plays the flag $\circlearrowleft$.

Gameplay

Example

- Let Π be the ASP program:

$$a \leftarrow \textbf{not } b$$

$$b \leftarrow \textbf{not } a$$

- O-Q1 can start with any literal in $\mathcal{L}$ but for our purposes will start with a?.
  He is effectively challenging P to prove a.
- P-Q2 responds $a \leftarrow \textbf{not } b$?
- O-Q3 responds b?
- P-Q4 – $b \leftarrow \textbf{not } a$?
- O-Q5 – a?

Gameplay continued

Example

- P-A5 – P has no moves left, and now answers with what he knows about a, which is **not** a
- O-A4 then answers P's question $b \leftarrow$ **not** a? with $\top$
- P-A3 must now play b as the rule $b \leftarrow$ **not** a? has been determined true in this play
- O-A2 must answer $\bot$
- P-A1 plays **not** a

P has played answers $\{$**not** $a, b\}$, which is an answer set of this program.

Problem I

Example

- Let Π be the ASP program:
$$\begin{array}{lll} a & \leftarrow & \textbf{not } b \\ b & \leftarrow & \textbf{not } a \\ & \leftarrow & b \end{array}$$
- O-Q1: starts with a?.
- P-Q2: responds $a \leftarrow \textbf{not } b$?
- O-Q3: responds b?
- P-Q4: $b \leftarrow \textbf{not } a$?
- O-Q5: – a?
- P-A5: **not** a
- O-A4: then answers P's question $b \leftarrow \textbf{not } a$? with $\top$

Problem II

Example

- P-A3 must now play b as the rule $b \leftarrow$ **not** a? has been determined true in this play
- O-A2: O then responds to $a \leftarrow$ **not** b? with $\bot$, as this rule isn't applied
- P-A1: then answers a? with **not** a
- Q-Q6: $\bot$?
- P-Q7: $\bot \leftarrow b$?
- O-Q8: O then queries b?
- P-A8: P knows b so answers b
- O-A7: $\top$
- now P cannot play $\bot$ (proving falsity) so loses

Backtracking

- Required: Without backtracking there is a deterministic polynomial algorithm for solving ASP programs.
  - We believe this is unlikely to be correct due to ASP being NP expressive.
- Complexity jump is partly due to presence of cycles in the programs with can have multiple evaluations:
- Example:

$$\begin{aligned} a &\leftarrow \textbf{not } b \\ b &\leftarrow \textbf{not } a \end{aligned}$$

- This cycle has two evaluations, $\{a, \textbf{not } b\}$ and $\{b, \textbf{not } a\}$.

Backtracking

- Let $s \in M^*$ and let K be a P-cycle in s – a set of open P-questions such that:
  - $K := \{m_{i_1}, \ldots, m_{i_k}\}$ and $head(m_{i_1}) \in body(m_{i_k})$
- Then let m_j be such that $\sigma(m_j) = m_{i_1}$. That relation still stands, but when in backtracking:
  - P plays the next move in K which has not been played from yet as the first move in a backtracking subplay.
- O cannot see any of the failed subsequences.

Backtracking Example I

Example

- Let Π be the ASP program:

$$\begin{array}{lll} a & \leftarrow & \textbf{not } b \\ b & \leftarrow & \textbf{not } a \\ & \leftarrow & b \end{array}$$

- O-Q1-c: $a?$.
- P-Q2-c: $a \leftarrow \textbf{not } b?$
- O-Q3-c: $b?$
- P-Q4-c: $b \leftarrow \textbf{not } a?$
- O-Q5-c: – $a?$
- P-A5-c: **not** a
- O-A4-c: $\top$
- P-A3-c: b

Backtracking Example II

Example

- O-A2-c: $\bot$
- P-A1-c: then answers $a?$ with **not** a
- Q-Q6: $\bot?$
- P-Q7: $\bot \leftarrow b?$
- O-Q8: $b?$
- P-A8: b
- O-A7: $\top$
- P-backtrack: $\circlearrowleft p$

Backtracking Example III

Example

- O-Q1-bc: responds *b*? New start of the cycle (instead of *a*?)
- P-Q2-bc: $b \leftarrow$ **not** *a*?
- O-Q3-bc: *a*?
- P-Q4-bc: $a \leftarrow$ **not** *b*?
- O-Q5-bc: *b*?
- P-A5-bc: **not** *b*
- O-A4-bc: $\top$
- P-A3-bc: *a*
- O-A2-bc: $\bot$
- P-A1-bc: **not** *b*

Backtracking Example IV

Example

- O-Q6-bc: $\bot$?
- P-Q7-bc: $\bot \leftarrow b$?
- O-Q8-bc: b?
- P-A8-bc: **not** b
- O-A7-bc: $\top$
- P-A6-bc: $\top$

Games, Programs and Answer Sets

Theorem

For Π an ASP program, $\sigma_\Pi := [\![\Pi]\!]$ is derived as previously and is winning.

Theorem

Let σ_Π be a (winning) strategy for the program Π. Let s be a play for σ_Π and $S = \{a \in HB_\Pi \mid a \in s\}$. Then, S is an answer set for Π.

Theorem

Let σ_Π be a (winning) strategy for the program Π and let S be an answer set for Π Then, there exists a play for σ_Π such that $S = \{a \in HB_\Pi \mid a \in s\}$. s

Applications

- Solvers.
  - The semantics here define automatically a notion of “On the fly” grounding.
- Verification:
  - The intensional nature of the semantics makes it easy to delve into the depths of algorithms:
    - Correctness (Debugging Algorithm – Brain, De Vos)

Conclusions

- We demonstrated that strategic and extensive game with perfect information can be modelled as variants of answer set programming such that the answer sets correspond with the equilibria of the games
- Game semantics can be used the characterise AnsProlog programs in a denotational way.