A foundational view of co-LP

Davide Ancona$^1$ and Agostino Dovier$^2$

$^1$Università di Genova
$^2$Università di Udine

CILC 2013
Catania
September 2013
Motivation

Several proposal by Gupta et al. on conductive logic programming, conductive logic programming with negation, conductive logic programming with constraints, applications of conductive logic programming, (from now on, simply co-LP)

...
Motivation

Several proposal by Gupta et al. on conductive logic programming, conductive logic programming with negation, conductive logic programming with constraints, applications of conductive logic programming, (from now on, simply co-LP)

... 

Some serious issues about the semantics

Some issues about the proposed co-SLD procedure

Some issues (easy to check) on the completeness of the interpreter

Some (inherited) issues about its correctness if negation is used
Outline

- Formal results on decidability for co-LP
- A simple operational semantics for co-LP
- Correctness based on the semantics of infinite tree LP (Jaffar, Stuckey)
- Completeness?
Let $P$ be a definite clause ground program and $I$ a set of atoms. Then

$$T_P(I) = \{ a : (a \leftarrow b_1, \ldots, b_n) \in P \land \{b_1, \ldots, b_n\} \subseteq I \}$$
Let $P$ be a definite clause ground program and $I$ a set of atoms. Then

$$T_P(I) = \{ a : (a \leftarrow b_1, \ldots, b_n) \in P \land \{ b_1, \ldots, b_n \} \subseteq I \}$$

$p.$
$q : - q.$
$r : - p, q, s.$
Let $P$ be a definite clause ground program and $I$ a set of atoms. Then

$$T_P(I) = \{ a : (a \leftarrow b_1, \ldots, b_n) \in P \land \{b_1, \ldots, b_n\} \subseteq I \}$$

$p.$
$q :\neg q.$
$r :\neg p, q, s.$

$$T_P(\emptyset) = \{ p \}.$$
Let $P$ be a definite clause ground program and $I$ a set of atoms. Then

$$T_P(I) = \{ a : (a \leftarrow b_1, \ldots, b_n) \in P \land \{b_1, \ldots, b_n\} \subseteq I \}$$

$p.$
$q : - q.$
$r : - p, q, s.$

$$T_P(\emptyset) = \{p\}, T_P(\{p\}) = \{p\} = lfp(T_P)$$
Let $P$ be a definite clause ground program and $I$ a set of atoms. Then

$$T_P(I) = \{ a : (a \leftarrow b_1, \ldots, b_n) \in P \land \{b_1, \ldots, b_n\} \subseteq I \}$$

$p.$
$q : - q.$
$r : - p, q, s.$

$T_P(\emptyset) = \{p\}, T_P(\{p\}) = \{p\} = \text{lfp}(T_P)$

$T_P(\{p, q, r, s\}) = \{p, q, r\}$
Let $P$ be a definite clause ground program and $I$ a set of atoms. Then

$$T_P(I) = \{ a : (a \leftarrow b_1, \ldots, b_n) \in P \land \{ b_1, \ldots, b_n \} \subseteq I \}$$

$p.$
$q :\leftarrow q.$
$r :\leftarrow p, q, s.$

$T_P(\emptyset) = \{ p \}, T_P(\{ p \}) = \{ p \} = \text{lfp}(T_P)$
$T_P(\{ p, q, r, s \}) = \{ p, q, r \}, T_P(\{ p, q, r \}) = \{ p, q \}$
Let $P$ be a definite clause ground program and $I$ a set of atoms. Then

$$T_P(I) = \{ a : (a \leftarrow b_1, \ldots, b_n) \in P \land \{b_1, \ldots, b_n\} \subseteq I \}$$

$p.$
$q :\leftarrow q.$
$r :\leftarrow p, q, s.$

$T_P(\emptyset) = \{p\}, T_P(\{p\}) = \{p\} = \text{lfp}(T_P)$

$T_P(\{p, q, r, s\}) = \{p, q, r\}, T_P(\{p, q, r\}) = \{p, q\},$
$T_P(\{p, q\}) = \{p, q\} = \text{gfp}(T_P)$
co-LP in a nutshell

Syntax

- Let us focus on the pure co-LP (Gupta et al. 1996)
- A co-LP program is a definite clause program.
- Namely a set of definite clauses

\[ A \leftarrow B_1, \ldots, B_n \]

where \( n \geq 0 \) and \( A \) and \( B_i \) are f.o. atomic formulas (atoms)

- The “standard” semantics of Logic Programming is based on \( \text{lfp}(T_P) \): a r.e. complete set, in general.
- The semantics of co-LP, instead is based on the greatest fix point
co-LP in a nutshell

Syntax

- Let us focus on the *pure* co-LP (Gupta et al. 1996)
- A co-LP program is a *definite clause program*.
- Namely a set of definite clauses

\[ A \leftarrow B_1, \ldots, B_n \]

where \( n \geq 0 \) and \( A \) and \( B_i \) are f.o. atomic formulas (atoms)

- The “standard” semantics of Logic Programming is based on \( \text{lfp}(T_P) \): a r.e. complete set, in general.
- The semantics of co-LP, instead is based on the greatest fix point
- By the way, since the idea is to capture perpetual processes, this fix point is computed on the extension of the Herbrand Universe that consider *infinite* terms, as well.
co-LP in a nutshell
Notions from Lloyd, 1987

- **complete Herbrand Universe** $\text{co-}U_P$: the set of finite and infinite terms built over functional symbols and variables
  - **rational terms**: can be represented by a *finite system of term equations*
    - Example: $\Omega = s(s(s(\cdots )))$ is represented by $X = s(X)$
  - **non rational terms**: cannot be represented by a finite system of term equations. Example: $[0, 1, 2, 3, \ldots ]$ ($0 = \emptyset, n + 1 = s(n)$)

- **complete Herbrand base** $\text{co-}B_P$: the set of all (possibly infinite, ground) atoms built on predicate symbols and terms in $\text{co-}U_P$

- **complete ground program** $\text{co-ground}(P)$: the set of all instances of clauses of $P$ where all variables are replaced by (possibly infinite) terms in $\text{co-}U_P$
co-LP in a nutshell

gfp-based semantics

- **model-theoretical semantics** of a definite clause program $P$
  - $T_P^{co} : \wp(\text{co-}B_P) \rightarrow \wp(\text{co-}B_P)$
    - $T_P^{co}(I) = \{ a : (a \leftarrow b_1, \ldots, b_n) \in \text{co-ground}(P) \land \{b_1, \ldots, b_n\} \subseteq I \}$
  - $P \models_{co} a \ (a \in \text{co-}B_P)$ if and only if $a \in \text{gfp}(T_P^{co})$
  - $P \models_{co} A \ (A \text{ atom possibly with variables})$ if and only if for all tree substitutions $\gamma : \text{FV}(A) \rightarrow \text{co-}U_P$, $P \models_{co} A_{\gamma}$ holds
**Iterated** $T_P^{co}$

\[
\begin{align*}
T_P^{co} \uparrow 0 & = \emptyset \\
T_P^{co} \uparrow \alpha & = T_P^{co} (T_P^{co} \uparrow (\alpha - 1)) \quad \text{if } \alpha \text{ is a successor ordinal} \\
T_P^{co} \uparrow \alpha & = \bigcup_{\beta < \alpha} T_P^{co} \uparrow \beta \quad \text{if } \alpha \text{ is a limit ordinal} \\
T_P^{co} \downarrow 0 & = \text{co-}B_P \\
T_P^{co} \downarrow \alpha & = T_P^{co} (T_P^{co} \downarrow (\alpha - 1)) \quad \text{if } \alpha \text{ is a successor ordinal} \\
T_P^{co} \downarrow \alpha & = \bigcap_{\beta < \alpha} T_P^{co} \downarrow \beta \quad \text{if } \alpha \text{ is a limit ordinal}
\end{align*}
\]

**Important property:** $\text{gfp}(T_P^{co}) = T_P^{co} \downarrow \omega$
Iterated $T^\text{co}_P$

\[
\begin{align*}
T^\text{co}_P \uparrow 0 &= \emptyset \\
T^\text{co}_P \uparrow \alpha &= T^\text{co}_P(T^\text{co}_P \uparrow (\alpha - 1)) \quad \text{if } \alpha \text{ is a successor ordinal} \\
T^\text{co}_P \uparrow \alpha &= \bigcup_{\beta < \alpha} T^\text{co}_P \uparrow \beta \quad \text{if } \alpha \text{ is a limit ordinal} \\
T^\text{co}_P \downarrow 0 &= \text{co-}B_P \\
T^\text{co}_P \downarrow \alpha &= T^\text{co}_P(T^\text{co}_P \downarrow (\alpha - 1)) \quad \text{if } \alpha \text{ is a successor ordinal} \\
T^\text{co}_P \downarrow \alpha &= \bigcap_{\beta < \alpha} T^\text{co}_P \downarrow \beta \quad \text{if } \alpha \text{ is a limit ordinal}
\end{align*}
\]

**Important property:** $gfp(T^\text{co}_P) = T^\text{co}_P \downarrow \omega$

**Remark 1:** this property does not hold for $T_P$ and finite terms

**Remark 2:** this property does not hold for $T^\text{co}_P$ if $\neq$ is allowed in the clauses
SLD with rational terms

Jaffar and Stuckey generalized SLD derivation — 1986 for Prolog II

Main ideas: unification without occurs check and use of a constraint store. For instance  
\[ P = p(X) \leftarrow p(s(X)). \]

\[ T_P \uparrow \omega = T_P^{co} \uparrow \omega = lfp(T_P^{co}) = \emptyset \]
\[ T_P^{co} \downarrow \omega = gfp(T_P^{co}) = co-B_P = \{ p(\Omega), p(0), p(1), p(2), p(3), \ldots \} \]

1. Infinite derivation for \( p(\Omega) \)

\[ \langle \{ X = s(X) \} \square p(X) \rangle \n \]
\[ \langle \{ X = s(X), X_1 = X \} \square p(s(X_1)) \rangle \n \]
\[ \langle \{ X = s(X), X_1 = X, X_2 = s(X_1) \} \square p(s(X_2)) \rangle \n \ldots \]

2. Infinite derivation for \( p(0) \)

\[ \langle \emptyset \square p(0) \rangle \n \]
\[ \langle \{ X_1 = 0 \} \square p(s(X_1)) \rangle \n \]
\[ \langle \{ X_1 = 0, X_2 = s(X_1) \} \square p(s(X_2)) \rangle \n \ldots \]
Operational semantics of co-LP
Gupta et. al. 2006

- Based on a state transition system which builds rational proof trees
- Example:

\[ \text{num}(s(X)) \leftarrow \text{num}(X). \]
\[ p(s(X)) \leftarrow \text{num}(X), p(s(X)). \]

Proof tree for \( p(\Omega) \):
Operational semantics of co-LP
Gupta et. al. 2006, formally

- A state is a pair \((T, E)\), where \(T\) is a finite tree with nodes labeled with atoms, and \(E\) is a system of term equations.
- A state \((T, E)\) transitions to another state \((T', E')\) by transition rule \(R\) of program \(P\) whenever:
  1. \(R\) is a definite clause of the form \(p(t'_0, \ldots, t'_n) \leftarrow B_1, \ldots, B_m\) and \(E' = \{t_1 = t'_1, \ldots, t_n = t'_n\} \cup E\) is solvable, and \(T'\) is obtained from \(T\) according to the following case analysis of \(m\):
     1. \(m = 0\) implies \(T'\) is obtained from \(T\) by removing a leaf labeled \(p(t_1, \ldots, t_n)\) and the maximum number of its ancestors, such that the result is still a tree.
     2. \(m > 0\) implies \(T'\) is obtained from \(T\) by adding children \(B_1, \ldots, B_m\) to a leaf labeled with \(p(t_1, \ldots, t_n)\).
  2. \(R\) is of the form \(\nu(m)\), a leaf \(N\) in \(T\) is labeled with \(p(t_1, \ldots, t_n)\), the proper ancestor of \(N\) at depth \(m\) is labeled with \(p(t'_1, \ldots, t'_n)\), \(E' = \{t_1 = t'_1, \ldots, t_n = t'_n\} \cup E\) is solvable, then \(T'\) is obtained from \(T\) by removing \(N\) and the maximum number of its ancestors, such that the result is still a tree.
Operational semantics of co-LP

Our proposal

- *hypothetical goal* (Bonatti, Pontelli, Son):
  \[ \langle E \square (A_1, S_1), \ldots, (A_n, S_n) \rangle, \text{ where } A_i \text{ are atoms and } S_i \text{ are the associated hypotheses (set of atoms)} \]

- derivation step from \( G = \langle E \square (A_1, S_1), \ldots, (A_n, S_n) \rangle \) to \( G' \) for \( P \):
  select atom \( A_i = p(s_1, \ldots, s_n) \), with hypotheses \( S_i \) and apply one of the following rules:

  1. let \( p(t_1, \ldots, t_n) \leftarrow B_1, \ldots, B_m \) be a renaming of a clause in \( P \) with fresh variables, and let \( E' = E \cup \{ s_1 = t_1, \ldots, s_n = t_n \} \) be solvable. Then \( G' = \langle E' \square (A_1, S_1), \ldots, (A_{i-1}, S_{i-1}), (B_1, S'), \ldots, (B_m, S'), (A_{i+1}, S_{i+1}), \ldots, (A_n, S_n) \rangle \)
     where \( S' = S_i \cup \{ p(s_1, \ldots, s_n) \} \).

  2. let \( p(t_1, \ldots, t_n) \in S_i \) be such that \( E' = E \cup \{ s_1 = t_1, \ldots, s_n = t_n \} \) is solvable. Then
     \( G' = \langle E' \square (A_1, S_1), \ldots, (A_{i-1}, S_{i-1}), (A_{i+1}, S_{i+1}), \ldots, (A_n, S_n) \rangle \).
Operational semantics of co-LP

**hypothesis goal** (Bonatti, Pontelli, Son):

\[ \langle E \sqcap (A_1, S_1), \ldots, (A_n, S_n) \rangle, \]

where \(A_i\) are atoms and \(S_i\) are the associated hypotheses (set of atoms)

**derivation step from** \(G = \langle E \sqcap (A_1, S_1), \ldots, (A_n, S_n) \rangle\) to \(G'\) for \(P\):

- select atom \(A_i = p(s_1, \ldots, s_n)\), with hypotheses \(S_i\) and apply one of the following rules:
  1. let \(p(t_1, \ldots, t_n) \leftarrow B_1, \ldots, B_m\) be a renaming of a clause in \(P\) with fresh variables, and let \(E' = E \cup \{s_1 = t_1, \ldots, s_n = t_n\}\) be solvable. Then
     \[ G' = \langle E' \sqcap (A_1, S_1), \ldots, (A_{i-1}, S_{i-1}), (B_1, S'), \ldots, (B_m, S'), (A_{i+1}, S_{i+1}), \ldots, (A_n, S_n) \rangle \]
     where \(S' = S_i \cup \{p(s_1, \ldots, s_n)\}\).
  2. let \(p(t_1, \ldots, t_n) \in S_i\) be such that \(E' = E \cup \{s_1 = t_1, \ldots, s_n = t_n\}\) is solvable. Then
     \[ G' = \langle E' \sqcap (A_1, S_1), \ldots, (A_{i-1}, S_{i-1}), (A_{i+1}, S_{i+1}), \ldots, (A_n, S_n) \rangle \].

- a SWI-Prolog meta-interpreter has been implemented directly from the 2 rules given above
Our proposal

Operational semantics of co-LP

Our proposal

\[\text{num}(s(X)) \leftarrow \text{num}(X).\]
\[p(s(X)) \leftarrow \text{num}(X), p(s(X)).\]

Example of successful derivation:

\[\langle \{X = s(X)\} \square (p(X), \emptyset) \rangle \vdash_{co}^\text{co}\]
\[\langle \{X = s(X), X = s(X_1)\} \square (\text{num}(X_1), \{p(X)\}), (p(s(X_1)), \{p(X)\}) \rangle \vdash_{co}^\text{co}\]
\[\langle \{X = s(X), X = s(X_1), X_1 = s(X_2)\} \square (\text{num}(X_2), \{p(X), \text{num}(X_1)\}), (p(s(X_1)), \{p(X)\}) \rangle \vdash_{co}^\text{co}\]
\[\langle \{X = s(X), X = s(X_1), X_1 = s(X_2), X_2 = X_1\} \square (p(s(X_1))), \{p(X)\}) \rangle \vdash_{co}^\text{co}\]
\[\langle \{X = s(X), X = s(X_1), X_1 = s(X_2), X_2 = X_1, s(X_1) = X\} \square \epsilon \rangle \vdash_{co}^\text{co}\]
Correctness

JS86 + “Pumping Lemma”

Let $P$ be a definite clause program. If there is a successful (hence finite) $\vdash_{\text{co}}$ derivation for $\langle E \Box (A, \emptyset) \rangle$ with c.a.s. $\theta$, then $P \models_{\text{co}} A\gamma$ for every term substitution $\gamma$ solution of $E\theta$.

proof sketch:

- if only rule 1 is applied, then the derivation is equivalent to a $\vdash_{\infty}$ derivation, and correctness directly follows from Jaffar and Stuckey results.
- if rule 2 is employed at least once, the proof is similar to that of the pumping lemma: a finite successful derivation can be transformed into an infinite derivation using only rule 1, which is, therefore, equivalent to a $\vdash_{\infty}$ derivation.
Decidability issues

\[ P = p(X) \leftarrow p(s(X)). \]

- \( T_P^{\text{co}} \downarrow \omega = \text{gfp}(T_P^{\text{co}}) = \text{co-}B_P \)
- the derivation for \( p(\Omega) \) is finite and successful
- the derivation for \( p(0) \) is infinite!
- is it possible to define a correct operational semantics for which there exists a finite successful derivation for \( p(0) \)?
Decidability issues

\[ P = p(X) \leftarrow p(s(X)). \]

- \( T_P^{co} \downarrow \omega = \text{gfp}(T_P^{co}) = \text{co-}B_P \)
- the derivation for \( p(\Omega) \) is finite and successful
- the derivation for \( p(0) \) is infinite!
- is it possible to define a correct operational semantics for which there exists a finite successful derivation for \( p(0) \)?
- Maybe, but unfortunately this is not possible in general!
Our proposal

Formal results on (un)decidability

- \( \Upsilon(S) \) denotes the subset of \( S \) containing only rational terms

**Theorem:**
- 1 \( \Upsilon(T^\omega_P) \) is recursively enumerable complete
- 2 \( \Upsilon(\text{co-}B_P \setminus T^\omega_P) \) is recursively enumerable complete (hence, \( \Upsilon(T^\omega_P) \) is productive).

Proof of (1): follows from known results.

Proof of (2): standard reduction from \( \bar{K} \) (building a suitable Prolog program s.t. \( x \in \bar{K} \) iff \( p(x) \in \text{gfp}(T^\omega_P) \))

---

Corollary: even when the semantics is restricted to rational terms, no complete procedure exists for establishing whether \( P |_{=co} a \); however, in absence of \( \neq \) symbols, there exists a complete procedure for establishing whether \( P \neq_{co} a \).
Formal results on (un)decidability

- $\gamma(S)$ denotes the subset of $S$ containing only rational terms

**Theorem:**
1. $\gamma(T_P^{co} \uparrow \omega)$ is recursively enumerable complete
2. $\gamma(\text{co-}B_P \setminus T_P^{co} \downarrow \omega)$ is recursively enumerable complete (hence, $\gamma(T_P^{co} \downarrow \omega)$ is productive).

**Proof of (1):** follows from known results.

**Proof of (2):** standard reduction from $\bar{K}$ (building a suitable Prolog program s.t. $x \in \bar{K}$ iff $p(x) \in \text{gfp}(T_P^{co})$)

**Corollary:** even when the semantics is restricted to rational terms, no complete procedure exists for establishing whether $P |_{co} a$; however, in absence of $\neq$ symbols, there exists a complete procedure for establishing whether $P \not{\models}_{co} a$. 
Our proposal

A famous picture

\[
lfp(T_P) \quad gfp(T_P) \quad T_P \downarrow \omega \quad B_P
\]
Example
Büchi $\omega$-automata

$\delta(s_0, a, s_1)$.
$\delta(s_1, b, s_2)$.
$\delta(s_2, c, s_3)$.
$\delta(s_2, e, s_0)$.
$\delta(s_3, d, s_0)$.
$\text{automata}([X|T], S) : -$ 
$\delta(S, X, S1)$,
$\text{automata}(T, S1)$. 
Example

Büchi $\omega$-automata

delta(s0, a, s1).
delta(s1, b, s2).
delta(s2, c, s3).
delta(s2, e, s0).
delta(s3, d, s0).

automata([X|T], S) :-
  delta(S, X, S1),
  automata(T, S1).

?- meta((automata(A, s0))).
A = [a, b, c, d|A] ;
A = [a, b, e|A] ;
A = [a, b, c, d, a, b, e|A] ;
A = [a, b, e, a, b, c, d|A] ;
...
Conclusions

Outline

- Formal results on decidability for co-LP
- A simple operational semantics for co-LP
- Correctness based on the semantics of infinite tree LP (Jaffar, Stuckey) + Pumping Lemma
- Completeness is impossible!
- Can be used for correctly detecting (some) properties
Conclusions

Outline

- Formal results on decidability for co-LP
- A simple operational semantics for co-LP
- Correctness based on the semantics of infinite tree LP (Jaffar, Stuckey) + Pumping Lemma
- Completeness is impossible!
- Can be used for correctly detecting (some) properties
- What about negation? And constraints?
Thank you

(We’re not selling co-LP, just explaining it)