Large-scale Cost-based Abduction in Full-fledged First-order Predicate Logic with Cutting Plane Inference

Naoya Inoue, Kentaro Inui

Communication Science Lab.
(Natural Language Processing Lab.)
Tohoku University, Japan

Cost-based Abduction (CBA)

Formally:
$H \cup B \models O$
$H \cup B \not\models \bot$

- Inference to the best explanation
  – Find the best reason (H) for what is observed (O), based on background knowledge (B)

Input: Observation

O = *get-gun(John) ^ go-to-store(John) ^ (∃x) rob(x)*

Background Knowledge

$$B = \begin{cases} (\forall x)\ hunt(x) \rightarrow get\text{-}gun(x) \\ (\forall x)\ go\text{-}shopping(x) \rightarrow go\text{-}to\text{-}store(x) \\ (\forall x)\ rob(x) \rightarrow get\text{-}gun(x) \\ (\forall x)\ rob(x) \rightarrow go\text{-}to\text{-}store(x) \end{cases}$$

Output: Best explanation

H_1 = *hunt(John) ^ go-shopping(John)*
H_2 = *rob(John)*
H_3 = *rob(John) ^ hunt(John) …*

Cost-based Abduction (CBA)

Formally:

$$H \cup B \models O$$
$$H \cup B \not\models \bot$$

- Inference to the best explanation
  - Find the best reason (H) for what is observed (O), based on background knowledge (B)

Input: Observation

$$O = \textit{get-gun}(John) \wedge \textit{go-to-store}(John) \wedge (\exists x)\, \textit{rob}(x)$$

Background Knowledge

$$B = \begin{cases} (\forall x)\, \textit{hunt}(x) \rightarrow \textit{get-gun}(x) \\ (\forall x)\, \textit{go-shopping}(x) \rightarrow \textit{go-to-store}(x) \\ (\forall x)\, \textit{rob}(x) \rightarrow \textit{get-gun}(x) \\ (\forall x)\, \textit{rob}(x) \rightarrow \textit{go-to-store}(x) \end{cases}$$

Output: Best explanation

$H_1 = \textit{hunt}(John) \wedge \textit{go-shopping}(John)$

$H_2 = \textit{rob}(John)$

$H_3 = \textit{rob}(John) \wedge \textit{hunt}(John) \ldots$

Cost-based Abduction (CBA)

Formally:

$$H \cup B \models O$$
$$H \cup B \not\models \bot$$

- Inference to the best (≡lowest-cost) explanation
  - Find the lowest-cost reason (H) for what is observed (O), based on background knowledge (B)

Input: Observation

$$O = \textit{get-gun}(John) \wedge \textit{go-to-store}(John) \wedge (\exists x)\, \textit{rob}(x)$$

Background Knowledge

$$B = \begin{cases} (\forall x)\, \textit{hunt}(x) \rightarrow \textit{get-gun}(x) \\ (\forall x)\, \textit{go-shopping}(x) \rightarrow \textit{go-to-store}(x) \end{cases}$$

How to evaluate explanations?
- Several *cost functions* have been proposed
- Basic criterion: "minimal explanation is favored"

Output: Best explanation

10.8 $H_1 = \textit{hunt}(John) \wedge \textit{go-shopping}(John)$
4.3 $H_2 = \textit{rob}(John)$
13.5 $H_3 = \textit{rob}(John) \wedge \textit{hunt}(John)$ …

Research Issue

- **Goal:** to model human language understanding with abduction
  - ✓ A large amount of world knowledge has become available as computational resources
  - ✓ Cost-based abduction would be a good solution to real-life natural language processing tasks [Hobbs+ 93, Ovchinnikova+ 11, etc.]
- **Issue:** how do we perform efficient inference with large knowledge bases (KBs) in first-order logic?
  - Most existing work targets “propositional” logic-based abduction
  - CBA is computationally expensive (combinatorial opt.)

CBA is computationally expensive

- Inference to the best explanation
  - Find the best reason (H) for what is observed (O), based on background knowledge (B)

Input: Observation

O = *get-gu… …ore(John)*

20 literals

- Mini-TACITUS (Mulkar-Mehta+ 07): ≥ **30 minutes**
- Markov Logic Networks (Richardson & Domingos 05)-based approach (Blythe+ 11): **7 minutes**

B = *go-… …store(X)*
rob(…) … get-gun(…)
rob(X) => go-to-store(X)

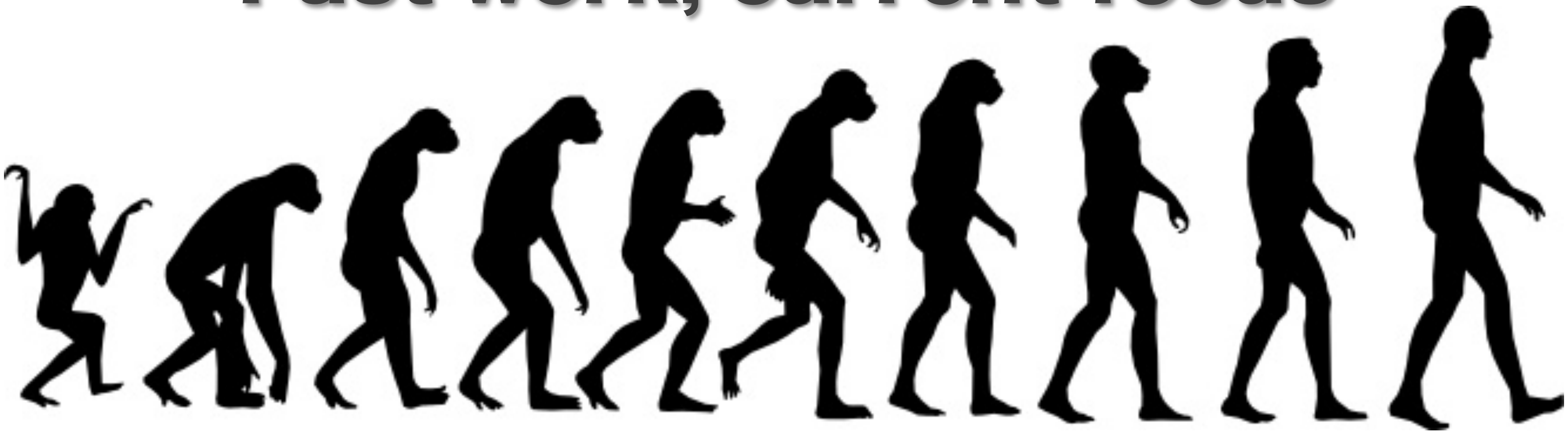
300,000+ axioms

Output: E…

Combinatorial optimization problem over 1,000 variables

H_3 = *rob(John) ^ hunt(John) …*

Past work, current focus



Work	Inference Method	Performance	Expressivity
Mulkar-Mehta 07	Brute forth	≥ 30 minutes	Subset of F.O.L
Blythe+ 11	Markov Logic Networks	7 minutes	Full F.O.L
Inoue & Inui 11	Integer Linear Programming (ILP)		
Inoue & Inui 12	ILP **+ Cutting Plane Inference**		

http://github.com/naoya-i/henry-n700/

- ✓ Introduction
- ☐ ILP-based approach to CBA
- ☐ Cutting plane inference for CBA
- ☐ Runtime evaluation

ILP-based approach to CBA

$$\operatorname*{arg\,min}_{H \in \mathcal{H}} cost(H)$$

- **Problem:** exponential growth of possible explanations $\mathcal{H}$
  - Naive strategy would not give a good solution in realistic time
- How do we find a better solution efficiently?
- **Key inspiration:**
  - CBA can be well-formulated through 0-1 ILP optimization problem
- **Solution:** exploit efficient search strategy developed in Operations Research fields (e.g. branch-and-bound) by formulating abduction as 0-1 ILP problem

Step 1. Search-space generation
Step 2. ILP optimization

Background knowledge B: Explanation H (Output): Observations O (Input):

$$B \cup H \models O$$

Best output ≡ lowest-cost:

$$\text{min. } cost(H)$$

ILP va

Step 1. Search-space generation

- Enumerate possible constituents of explanations

Step 2. ILP optimization

- Find the best combination of the constituents based on cost function

Step 1. Search-space generation
Step 2. ILP optimization

Background knowledge B:

$(\forall x)\ hunt(x) \rightarrow get\text{-}gun(x)$
$(\forall x)\ go\text{-}shopping(x) \rightarrow go\text{-}to\text{-}store(x)$
$(\forall x)\ rob(x) \rightarrow get\text{-}gun(x)$
$(\forall x)\ rob(x) \rightarrow go\text{-}to\text{-}store(x)$

$\cup$

Explanation H (Output):

$\models$

Observations O (Input):

$get\text{-}gun(John)$
$go\text{-}to\text{-}store(John)$
$(\exists x)\ rob(x)$

Potential Elemental Hypotheses (P):

Step 1-1: enumerate set of literals that can entail (part of) observations. Explanation (output) is represented by combination of these literals.

Best output ≡ lowest-cost:

$$\min.\ cost(H)$$

ILP variables:

ILP objective:

Step 1. Search-space generation
Step 2. ILP optimization

Background knowledge B:

$(\forall x)\ hunt(x) \rightarrow get\text{-}gun(x)$
$(\forall x)\ go\text{-}shopping(x) \rightarrow go\text{-}to\text{-}store(x)$
$(\forall x)\ rob(x) \rightarrow get\text{-}gun(x)$
$(\forall x)\ rob(x) \rightarrow go\text{-}to\text{-}store(x)$

$\cup$

Explanation H (Output):

$\models$

Observations O (Input):

$get\text{-}gun(John)$
$go\text{-}to\text{-}store(John)$
$(\exists x)\ rob(x)$

Potential Elemental Hypotheses (P):

$get\text{-}gun(John)$ $go\text{-}to\text{-}store(John)$ $\exists x\ rob(x)$
$hunt(John)$ $go\text{-}shopping(John)$ $rob(John)$
$x = John$

Best output ≡ lowest-cost:

$$\min.\ cost(H)$$

ILP variables:

rob(John)* ∧ *rob(x)* ∧ *x=John
yields the smaller hypothesis: ***rob(John)***

Step 1. Search-space generation
Step 2. ILP optimization

Background knowledge B:

$(\forall x)\ hunt(x) \rightarrow get\text{-}gun(x)$

$(\forall x)\ go\text{-}shopping(x) \rightarrow go\text{-}to\text{-}store(x)$

$(\forall x)\ rob(x) \rightarrow get\text{-}gun(x)$

$(\forall x)\ rob(x) \rightarrow go\text{-}to\text{-}store(x)$

∪

Explanation H (Output):

$get\text{-}gun(John)$

$go\text{-}to\text{-}store(John)$

Observations O (Input):

$get\text{-}gun(John)$

$go\text{-}to\text{-}store(John)$

(x)

Step 2-3: introduce ILP constraints:
- Entailment relations

Step 2-3: introduce ILP constraints:
- Transitivity over equalities

Potential Elemental Hypotheses (P):

$get\text{-}gun(John)$ $go\text{-}to\text{-}store(John)$ $\exists x\ rob(x)$

$hunt(John)$ $go\text{-}shopping(John)$ $rob(John)$

$x = John$

Best output ≡ lowest-cost:

$$\text{min. } cost(H) = \sum_{p \in \{p \mid p \in H,\, p \text{ is not entailed}\}} cost(p)$$

ILP variables:

Step 2-1: assign 0-1 ILP variables h or s to each potential elemental hypothesis.

h_p: 1 if p is included in explanation

$s_{x,y}$: 1 if x and y are unified in explanation

ILP objective:

$$\text{min. } cost(H)$$

= **Step 2-2:** represent cost function using 0-1 ILP variables.

Step 1. Search-space generation
Step 2. ILP optimization

Background knowledge B:

$(\forall x)\ hunt(x) \rightarrow get\text{-}gun(x)$
$(\forall x)\ go\text{-}shopping(x) \rightarrow go\text{-}to\text{-}store(x)$
$(\forall x)\ rob(x) \rightarrow get\text{-}gun(x)$
$(\forall x)\ rob(x) \rightarrow go\text{-}to\text{-}store(x)$

Explanation H (Output):

∪

Observations O (Input):

$get\text{-}gun(John)$
$go\text{-}to\text{-}store(John)$
(x)

Entailment relations
e.g. $h_{hunt(John)} \geq h_{get\text{-}gun(John)}$

Transitivity over equalities
e.g.:
$s_{x=John} + s_{y=John} - 1 \leq s_{x=y}$
$s_{y=John} + s_{x=y} - 1 \leq s_{x=John}$
$s_{x=y} + s_{x=John} - 1 \leq s_{y=John}$

Potential Elem

$get\text{-}gun(John)$
$hunt(John)$ $go\text{-}shopping(John)$ $rob(John)$
$x = John$

≡ lowest-cost:

$$t(H) = \sum_{p \in \{p \mid p \in H,\ p \text{ is not explained}\}} cost(p)$$

ILP variables:

$h_{get\text{-}gun(John)}$ $h_{go\text{-}to\text{-}store(John)}$ $h_{\exists x\ rob(x)}$
$h_{hunt(John)}$ $h_{go\text{-}shopping(John)}$ $h_{rob(John)}$
$s_{x,John}$

ILP objective:

$$\min.\ cost(H) = \sum_{p \in P} [h_p \cdot cost(p) - r_p \cdot cost(p)]$$

Step 1. Search-space generation
Step 2. ILP optimization

Background knowledge B:

$(\forall x)\ hunt(x) \rightarrow get\text{-}gun(x)$
$(\forall x)\ go\text{-}shopping(x) \rightarrow go\text{-}to\text{-}store(x)$
$(\forall x)\ rob(x) \rightarrow get\text{-}gun(x)$
$(\forall x)\ rob(x) \rightarrow go\text{-}to\text{-}store(x)$

Explanation H (Output):

Observations O (Input):

$get\text{-}gun(John)$
$go\text{-}to\text{-}store(John)$

Entailment relations
e.g. $h_{hunt(John)} \geq h_{get\text{-}gun(John)}$

Transitivity over equalities
e.g.:
$s_{x=John} + s_{y=John} - 1 \leq s_{x=y}$
$s_{y=John} + s_{x=y} - 1 \leq s_{x=John}$
$s_{x=y} + s_{x=John} - 1 \leq s_{y=John}$

Potential Elem

$get\text{-}gun(John)$
$hunt(John)$ $go\text{-}shopping(John)$ $rob(John)$
$x = John$ $rob(John)$

lowest-cost:

$$= \sum_{p \in \{p \mid p \in H,\, p \text{ is not explained}\}} cost(p)$$

ILP variables:

$h_{get\text{-}gun(John)}$ $h_{go\text{-}to\text{-}store(John)}$ $h_{\exists x\, rob(x)}$
$h_{hunt(John)}$ $h_{go\text{-}shopping(John)}$ $h_{rob(John)}$
$s_{x,John}$

ILP objective:

$$\min.\ cost(H) = \sum_{p \in P} [h_p \cdot cost(p) - r_p \cdot cost(p)]$$

- ✓ Introduction
- ✓ ILP-based approach to CBA
- ☐ Cutting plane inference for CBA
- ☐ Runtime evaluation

Weak point of ILP-based approach

- The number of transitivity constraints over equality relations grows cubically

$$\text{for all logical variables } x, y, z:$$
$$x{=}y \wedge y{=}z \Rightarrow x{=}z \ (s_{x,y} + s_{y,z} - 1 \le s_{x,z})$$
$$y{=}z \wedge x{=}z \Rightarrow x{=}y \ (s_{y,z} + s_{x,z} - 1 \le s_{x,y})$$
$$x{=}z \wedge x{=}y \Rightarrow y{=}z \ (s_{x,z} + s_{x,y} - 1 \le s_{y,z})$$

  – Order: (the number of logical variables)^3
- Processing time quickly increases when observations and/or knowledge base are large
- How can we reduce the computational complexity?

Cutting Plane Inference (CPI)

- Iterative optimization strategy for solving optimization problems with large/infinite constraints [Dantzig+ 54]
  - Returns exact solution
- Applied for various optimization problems:
  - Parameter estimation in machine-learning [Joachims & Finley 09, etc.]
  - Structured prediction problems [Riedel 06, 08, etc.]
- The algorithm:
  - “all the constraints might not be violated at once”

$C \leftarrow \{\}$: set of constraints
repeat
  optimize with C
  add violated constraints to C
until C does not change

Cutting Plane Inference for CBA

- Solution to cubic growth of transitivity constraints!
  - "all the transitivity constraints might not be violated at once"

General CPI:

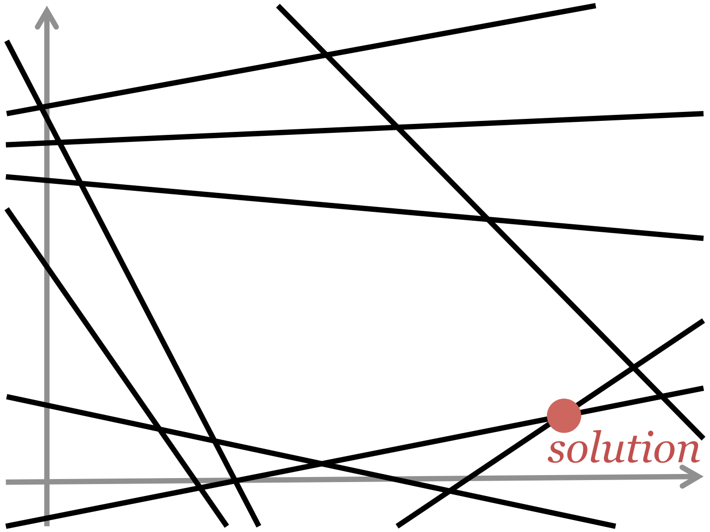
$C \leftarrow \{\}$: set of constraints
repeat
  optimize with C
  add violated constraints to C
until C does not change

CPI for CBA:

$C \leftarrow \{\}$: set of transitivity constraints
repeat
  perform CBA with C
  add violated transitivity constraints to C
until C does not change

- Benefits:
  - Not required to generate all the transitivity constraints in advance
  - Much greater chance to get suboptimal ILP solutions
  - The overall inference time might be faster

Working example



Explanation H (Output):

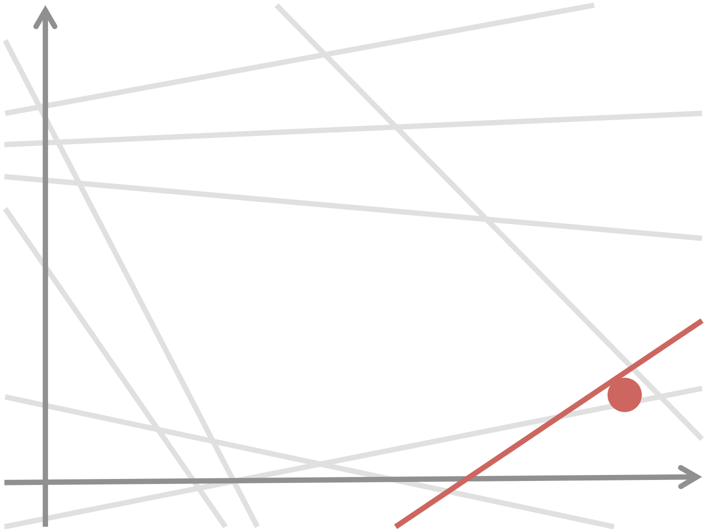
Transitivity constraints that should be satisfied

Actually concerned constraints (C)

$x=y \wedge y=z \Rightarrow x=z$
$y=z \wedge x=z \Rightarrow x=y$
$x=z \wedge x=y \Rightarrow y=z$
$x=y \wedge y=w \Rightarrow x=w$
$y=w \wedge x=w \Rightarrow x=y$
$x=w \wedge x=y \Rightarrow y=w$
$x=z \wedge z=w \Rightarrow x=w$
$z=w \wedge x=w \Rightarrow x=z$
:

$C \leftarrow \{\}$: set of transitivity constraints
repeat
perform CBA with C
add violated transitivity constraints to C
until C does not change

Working example



Explanation H (Output):

$(\exists x)\ p(x)$
$(\exists y)\ p(y)$
$(\exists z)\ p(z)$
$q(A)$
$y=z$
$x=z$

Transitivity constraints that should be satisfied	Actually concerned constraints (C)

$x=y \wedge y=z \Rightarrow x=z$

violated $\boxed{y=z \wedge x=z \Rightarrow x=y}$ → $y=z \wedge x=z \Rightarrow x=y$

$x=z \wedge x=y \Rightarrow y=z$

$x=y \wedge y=w \Rightarrow x=w$

$y=w \wedge x=w \Rightarrow x=y$

$x=w \wedge x=y \Rightarrow y=w$

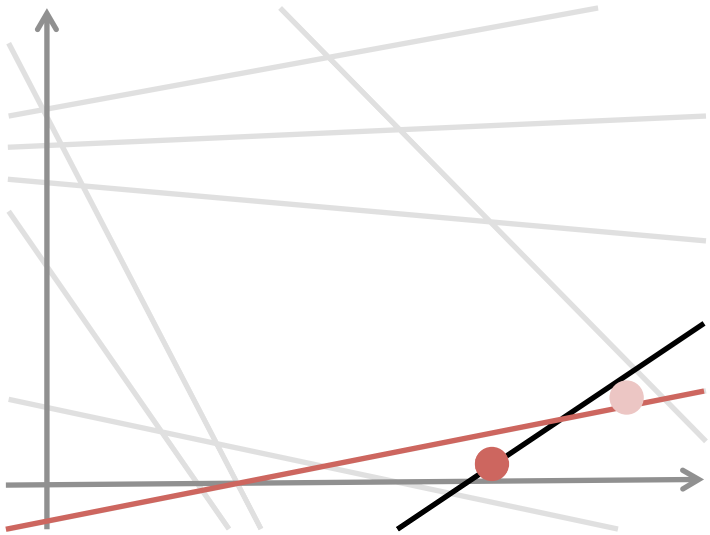
$x=z \wedge z=w \Rightarrow x=w$

$z=w \wedge x=w \Rightarrow x=z$

:

$C \leftarrow \{\}$: set of transitivity constraints
repeat
▶ perform CBA with C
▶ add violated transitivity constraints to C
until C does not change

Working example



Explanation H (Output):

$(\exists x)\ p(x)$
$(\exists y)\ p(y)$
$(\exists z)\ p(z)$
$q(A)$
$z{=}w$
$x{=}w$

Transitivity constraints that should be satisfied

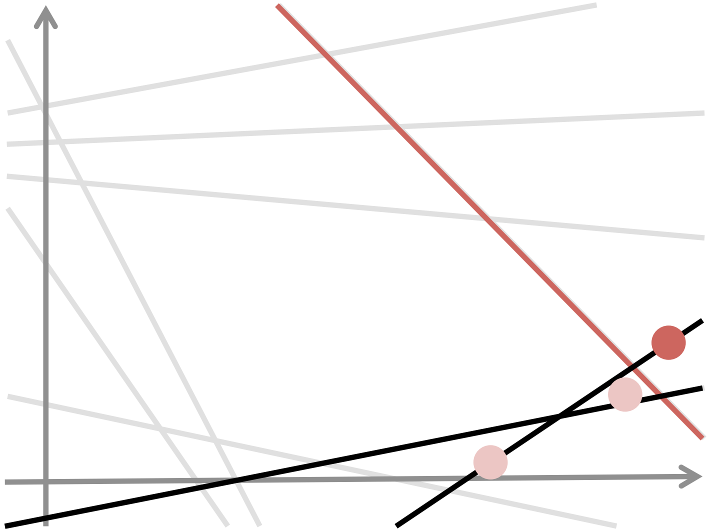
$x{=}y \wedge y{=}z \Rightarrow x{=}z$
$y{=}z \wedge x{=}z \Rightarrow x{=}y$
$x{=}z \wedge x{=}y \Rightarrow y{=}z$
$x{=}y \wedge y{=}w \Rightarrow x{=}w$
$y{=}w \wedge x{=}w \Rightarrow x{=}y$
$x{=}w \wedge x{=}y \Rightarrow y{=}w$
$x{=}z \wedge z{=}w \Rightarrow x{=}w$
violated $\boxed{z{=}w \wedge x{=}w \Rightarrow x{=}z}$
:

Actually concerned constraints (C)

$y{=}z \wedge x{=}z \Rightarrow x{=}y$
$z{=}w \wedge x{=}w \Rightarrow x{=}z$

$C \leftarrow \{\}$: set of transitivity constraints
repeat
▶ perform CBA with C
▶ add violated transitivity constraints to C
until C does not change

Working example



Explanation H (Output):

$(\exists x)\ p(x)$
$(\exists y)\ p(y)$
$(\exists z)\ p(z)$
$q(A)$
$x{=}z$
$z{=}w$

Transitivity constraints that should be satisfied

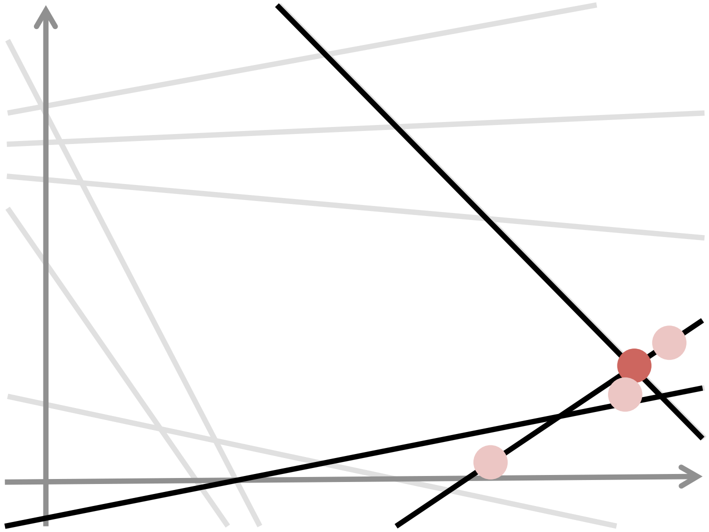
$x{=}y \wedge y{=}z \Rightarrow x{=}z$
$y{=}z \wedge x{=}z \Rightarrow x{=}y$
$x{=}z \wedge x{=}y \Rightarrow y{=}z$
$x{=}y \wedge y{=}w \Rightarrow x{=}w$
$y{=}w \wedge x{=}w \Rightarrow x{=}y$
$x{=}w \wedge x{=}y \Rightarrow y{=}w$
violated $\boxed{x{=}z \wedge z{=}w \Rightarrow x{=}w}$
$z{=}w \wedge x{=}w \Rightarrow x{=}z$
:

Actually concerned constraints (C)

$y{=}z \wedge x{=}z \Rightarrow x{=}y$
$z{=}w \wedge x{=}w \Rightarrow x{=}z$
$x{=}z \wedge z{=}w \Rightarrow x{=}w$

$C \leftarrow \{\}$: set of transitivity constraints
repeat
▶ perform CBA with C
▶ add violated transitivity constraints to C
until C does not change

Working example



Explanation H (Output):

$(\exists x)\ p(x)$
$(\exists y)\ p(y)$
$(\exists z)\ p(z)$
$q(A)$
$y=z$
$x=z$
$x=y$

Transitivity constra

No violations!

Optimal solution can be found with just 3 constraints (originally 12).

concerned constraints (C)

$x=y$ ^

$y=z$ ^

$x=z$ ^ $x=y \Rightarrow y=z$

$x=y$ ^ $y=w \Rightarrow x=w$

$y=w$ ^ $x=w \Rightarrow x=y$

$x=w$ ^ $x=y \Rightarrow y=w$

$x=z$ ^ $z=w \Rightarrow x=w$

$z=w$ ^ $x=w \Rightarrow x=z$

:

^ $x=z \Rightarrow x=y$

w ^ $x=w \Rightarrow x=z$

$x=z$ ^ $z=w \Rightarrow x=w$

$C \leftarrow \{\}$: set of transitivity constraints
repeat
perform CBA with C
add violated transitivity constraints to C
until C does not change

- ✓ Introduction
- ✓ ILP-based approach to CBA
- ✓ Cutting plane inference for CBA
- □ Runtime evaluation

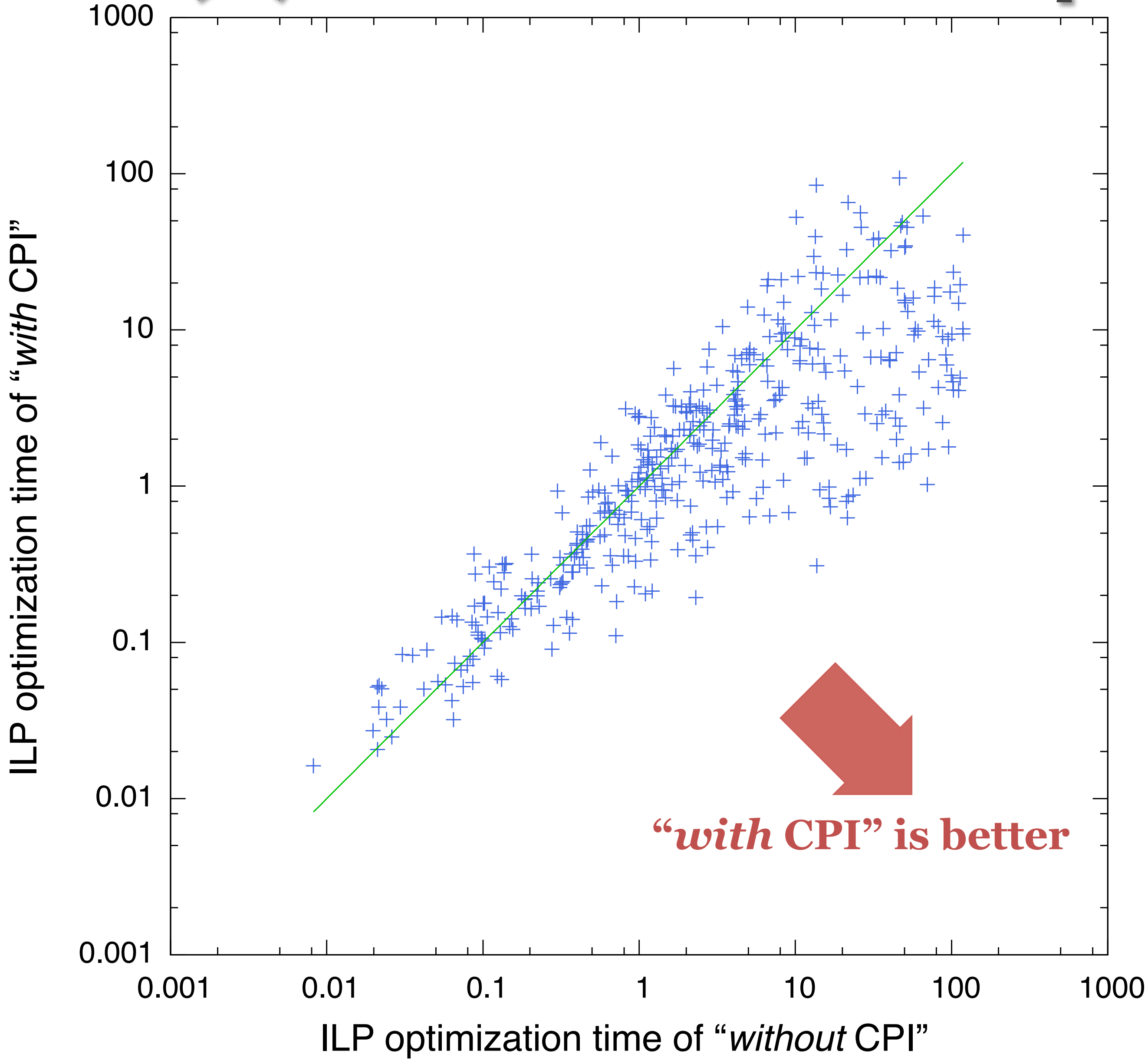
Runtime evaluation

- How much does CPI improve the *inference time*?
- Dataset
  - Input: Recognizing Textual Entailment (RTE) [Dagan et al. 09]
    - Real-life task in natural language processing
      - Given two texts T and H, predict whether T entails H or not
      - e.g. T: John tangoed. / H: John danced.
    - Texts are converted to logical forms through Boxer [Bos 09]
      - 30 literals on average x 800 problems
  - Background knowledge: 300,000 axioms from popular lexical databases [Fellbaum 98, Baker 98]
    - 289,655 axioms from WordNet (e.g. synset9(x) => synset10(x))
    - 7,558 axioms from FrameNet
      (e.g. GIVING(e1, x, y) => GETTING(e2, y, z))
- Tool
  - ILP solver: Gurobi optimizer 5.0

Results (1): effects on average time

- Given time limit = 120 sec.,

Results (2): effects on each problem



Summary

- A large amount of world knowledge has become available as computational resources, which makes CBA a promising solution to natural language processing tasks
- Proposed CPI-based approach to scale up cost-based abduction to larger problems
- CPI-based approach significantly improved both search-space generation and ILP inference runtimes on large dataset
- The inference engine is publicly available:

http://github.com/naoya-i/henry-n700/

Ongoing/future work

- Ongoing work
  - Evaluate on real-life natural language processing tasks
    - Ongoing: RTE, coreference resolution (result: "*not bad...*")
  - Developing machine learning of costs
    - Integration of *statistical machine learning* and *logical inference*
    - CPI enables a learning mechanism to work (inference is a subroutine of learning)
- Future: apply CPI to search-space generation
  - Not enumerate potential elemental hypotheses in advance
  - Repeat:
    - (i) accumulating potential elemental hypotheses that would give the best explanation (according to some score function)
    - (ii) peforming CBA on the accumulated set
  - Analogously to CPI for Markov Logic Networks [Riedel 08]