# Learning Deduction Rules by Induction

**Chiaki Sakama** (Wakayama University)
**Tony Ribeiro** (Graduate Univ. for Advanced Studies)
**Katsumi Inoue** (National Institute of Informatics)

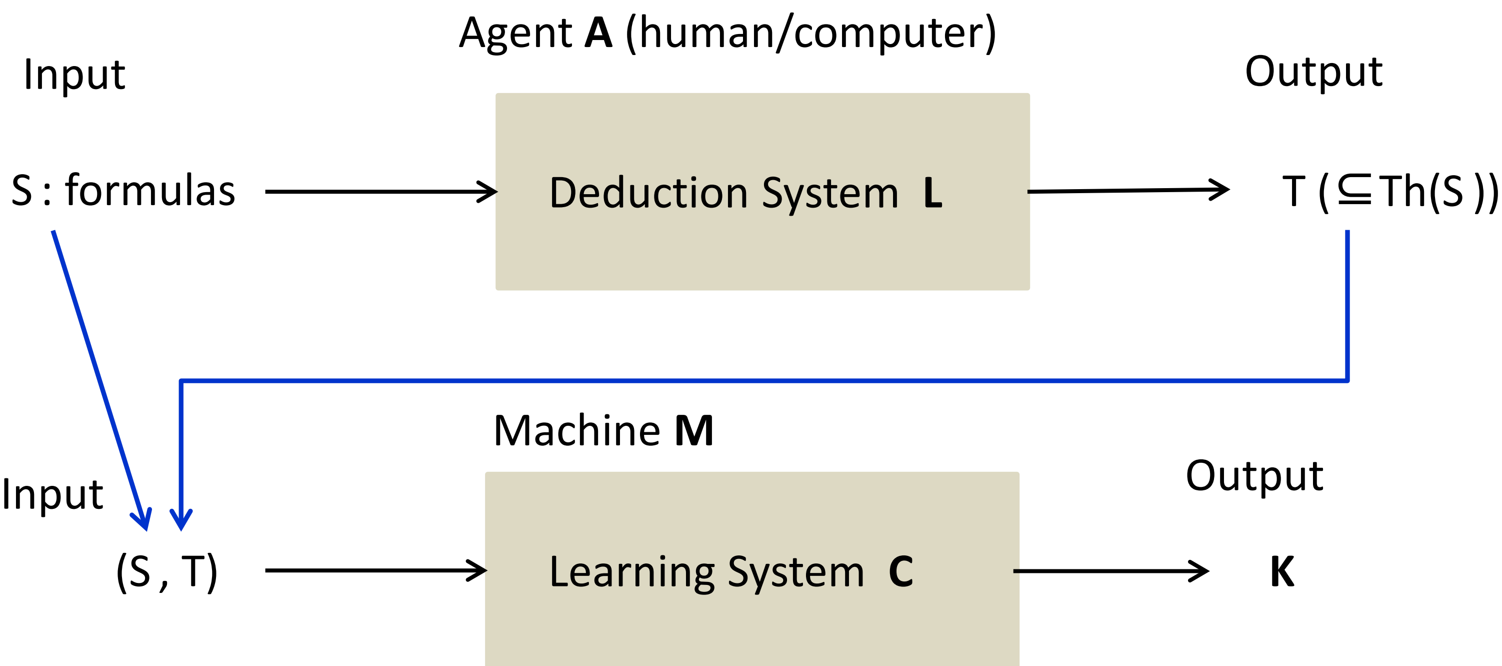
**ILP2015, Kyoto**

# Leaning Logics

- **Given**: a set **S** of formulas and their logical consequences **T**.
- **Find**: an axiomatic system **K** that produces **T** from **S**.

# “Can Machines Learn Logics?”

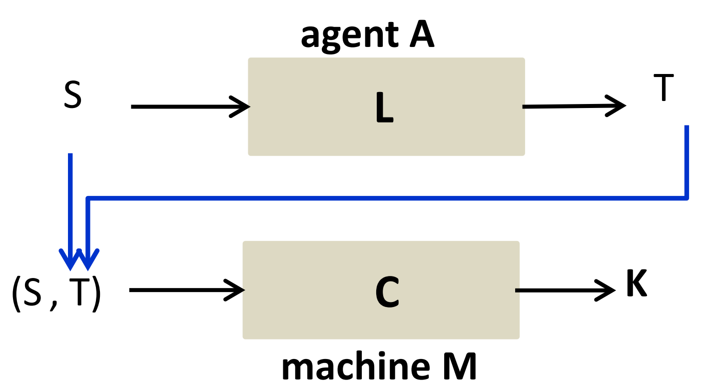
(C. Sakama & K. Inoue, 8th Int’l Conf. Artificial General Intelligence, Berlin, July 2015; LNAI 9205)




Given input (S, T), a machine **M** produces an axiomatic system **K**.

# Challenging Problems

- Can we develop an algorithm **C** for learning a classical or non-classical logic **L**?
- Does a machine **M** discover a **new** axiomatic system **K** such that **K** |- F iff **L**|- F for any formula F?



# Outline of this study

- We use the **LF1T** induction algorithm (learning from 1-step transitions) for learning deduction rules of propositional logic.
- We show some experimental results.

# LF1T: Learning from 1-step transitions

(Inoue, Ribeiro, Sakama, **MLJ** 2014)†

- **Input**: a set **E** of pairs of (Herbrand) interpretations
- **Output**: a program $P$ s.t. $J = T_P(I)$ for any $(I, J) \in$ **E**

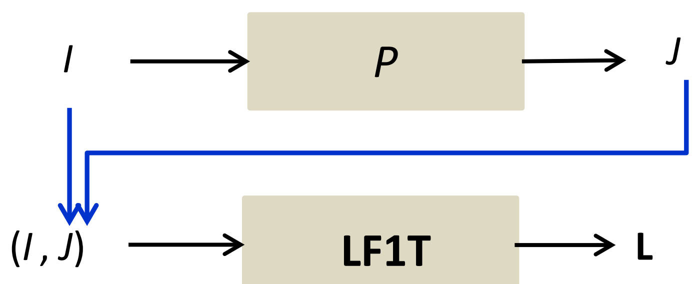
where

- $P$ is a (propositional) definite logic program
- $T_P(I)=\{ a \mid a \leftarrow b_1,\ldots,b_n$ is in $P$ s.t. $\{b_1,\ldots,b_n\} \subseteq I \}$
- A rule $a \leftarrow b_1,\ldots,b_n$ is **consistent** with $(I, J)$ if $\{b_1,\ldots,b_n\} \subseteq I$ implies $a \in J$

Note: **LF1T** is introduced for normal logic programs in (†)

# Learning Deduction Rules by LF1T

- We assume a deduction system **L** represented by a **metalogic program** *P* that provides transitions (*I*, *J*) satisfying $J = T_P(I)$.
- Given (*I*, *J*) as an input, our goal is to examine whether **LF1T** can reproduce correct inference rules of **L** represented by meta-rules in *P*.



# Example

- Given the Herbrand base:

  **B**={ hold(p), hold(q), hold(r), hold(p→r) },

  a rule with hold(r) in the head is constructed as follows.
- **Step 0**: **LF1T** starts with the most general rule:

  hold(r)← (1)
- **Step 1**: The transition ({},{}) is given. (1) is inconsistent with this (namely, {} should produce {hold(r)} under (1)), so (1) is minimally specialized by introducing an atom from **B**:

  hold(r)←hold(p) (2)
  hold(r)←hold(q) (3)
  hold(r)←hold(r) (4)
  hold(r)←hold(p→r) (5)

# Example

- **Step 2**: The transition ({hold(p)},{hold(p)}) is given.

  hold(r)←hold(p) (2)

  is inconsistent with this, so (2) is specialized into

  hold(r)←hold(p), hold(q)

  hold(r)←hold(p), hold(r)

  hold(r)←hold(p), hold(p→r)

  These rules are respectively subsumed by

  hold(r)←hold(q) (3)

  hold(r)←hold(r) (4)

  hold(r)←hold(p→r) (5)

  hence removed. As a result, (3),(4) and (5) remain.

# Example

- **Step 3**: The transition ({hold(q)},{hold(q)}) is given.

$$hold(r) \leftarrow hold(q) \qquad (3)$$

is inconsistent with this, so (3) is specialized into

$$hold(r) \leftarrow hold(q), hold(p) \qquad (6)$$
$$hold(r) \leftarrow hold(q), hold(r)$$
$$hold(r) \leftarrow hold(q), hold(p \rightarrow r)$$

The last two rules are respectively subsumed by

$$hold(r) \leftarrow hold(r) \qquad (4)$$
$$hold(r) \leftarrow hold(p \rightarrow r) \qquad (5)$$

and removed. As a result, (4), (5) and (6) remain.

| input | output |
| --- | --- |
| ({}, {}) | hold(r)←hold(p) hold(r)←hold(q)<br>hold(r)←hold(r) hold(r)←hold(p→r) |
| ({hold(p)},{hold(p)}) | ~~hold(r)←hold(p)~~ hold(r)←hold(q)<br>hold(r)←hold(r) hold(r)←hold(p→r) |
| ({hold(q)},{hold(q)}) | ~~hold(r)←hold(q)~~ hold(r)←hold(r)<br>hold(r)←hold(p→r) hold(r) ←hold(p),hold(q) |
| ({hold(p→r)},{hold(p→r)}) | hold(r)←hold(r) ~~hold(r)←hold(p→r)~~<br>hold(r) ←hold(p),hold(q)<br>hold(r)←hold(p→r),hold(p)<br>hold(r)←hold(p→r),hold(q) |
| ({hold(p),hold(q)},{hold(p),hold(q)}) | hold(r)←hold(r) ~~hold(r) ←hold(p),hold(q)~~<br>hold(r)←hold(p→r),hold(p)<br>hold(r)←hold(p→r),hold(q) |
| ({hold(p→r),hold(q)},{hold(p→r),hold(q)}) | hold(r)←hold(r)<br>hold(r)←hold(p→r),hold(p)<br>~~hold(r)←hold(p→r),hold(q)~~ |
| ({hold(p→r),hold(p)}, {hold(p),hold(r)}) | hold(r)←hold(r) :Repetition<br>hold(r)←hold(p→r),hold(p) :Modus Ponens |

# Experimental Results

- Given **B**={ hold(p), hold(¬p), hold(q), hold(¬q), hold(p→q), hold(q→r), hold(p→r) }, **LF1T** produces:
  - hold(¬p) ← hold(¬q) ∧ hold(p→q) : **Modus Tollens**
  - hold(p→r) ← hold(p→q) ∧ hold(q→r) : **Hypothetical Syllogism**
- Given **B**={ hold(p), hold(¬p), hold(q), hold(¬q), hold(p∨q), hold(¬p∨¬q), hold(r∨s), hold(¬r∨¬s), hold(p→r), hold(q→s) }, **LF1T** produces:
  - hold(p) ← hold(p∨q) ∧ hold(¬q) : **Disjunctive Syllogism**
  - hold(r∨s) ← hold(p∨q) ∧ hold(p→r) ∧ hold(q→s) : **Constructive Dilemma**
  - hold(¬p∨¬q) ← hold(¬r∨¬s) ∧ hold(p→r) ∧ hold(q→s) : **Destructive Dilemma**
- **Given a transition (I,J)=**({hold(p→q), hold(q)}, {hold(p)}), **LF1T** produces
  - hold(p) ← hold(q)∧hold(p→q) : **Fallacy of Affirming the Consequence** (a rule for **Abduction**)

# Summary

- Given transitions specifying premises and their consequences, **LF1T** successfully produces inference rules of natural deduction in propositional logic.
- The method is applied to learning non-deductive inference rules such as abduction.
- A limitation is that the number of possible transitions increases exponentially in proportion to the size of the Herbrand base.