Yogesh R. Isawe

# $i^2$ Learning : Overcoming Mismatching & Asymmetric Inconsistencies

**Abstract:**

Developing stimulus-specific inconsistency learning algorithms and heuristics for the framework of perpetual learning agents for; 1) Asymmetric inconsistencies & 2) Mismatching inconsistencies

Inconsistencies in an agent's decision-making indicates that the knowledge base of agent can't properly and adequately handle encountered inconsistent circumstances.

Thus when the agent encounters a conflicting circumstance for which it does not know how to handle, it needs to initiate the learning episode that relies on the inconsistency-specific heuristics to revise, refine, or augment existing knowledge to adapt to the emergent patterns and behaviors.

Scope:

A) Developing the inconsistency-specific learning algorithms and heuristics for the inconsistency-induced learning framework which has been proposed in the previous related work in the i2Learning.

B) Developing and defining these stimulus-specific learning strategies and algorithms for the pertinent domain specific real world application.

## Machine Learning:

The primary objective of intelligent agent systems is problem solving, learning is just the means for agents to get progressively better at what they do. When an agent system can adequately handle its tasks, learning may not be needed.

Definition:

The machine learning is defined using three first-class entities: T, P, and E. A machine learns with respect to a particular task T, performance metric P, and type of experience E; if the system reliably improves its performance P at task T, following an experience E.

The general practice in the field of machine learning is still dominated by the one-time learner approach, where some learning algorithm is utilized on data sets to produce certain model or target function, and then the learner is put away and the model or function is put to work. Such a learn-once-apply-

next (or LOAN) approach is not adequate in dealing with many real world problems and is in sharp contrast with the human's lifelong learning process.

**Need for Perpetual Learning:**

Most of the agent systems do not come equipped with a complete set of problem solving knowledge or data set.

Learning algorithm in LOAN is used to produce a model or a target function *h* that is put to work at task T. There is no continual refinement or augmentation of the problem-solving knowledge embodied in the initially learned *'h' (model or target function)* so as to incrementally and continuously improves performance P of the problem solver at future variations of task T.

When the agent encounters, during problem solving, a conflicting circumstance for which it does not know how to handle, it needs to pause in its current problem-solving episode, and switches to a learning episode through which the agent acquires necessary knowledge for resolving the circumstance.

Since the primary objective of intelligent agent systems is problem solving, the agent should returns to its problem solving routine until it is pushed to the edge of its knowledge by other learning stimuli. This type of perpetual learning process enables agents to incrementally and continuously improve their problem-solving performance over time.

**Perpetual machine learning:**

Human life-long learning process is often triggered by events, stimuli, desires, goals or objectives. The human learning process is episodic, a learning agent can also rely on a set of stimuli for its successive learning episodes.

To define such perpetual learning agent system, we need to elevate learning stimuli of first class, by adding a set of learning stimuli S to the first-class status such that perpetual learning can be formulated in terms of (T, P, E, and S).

Definition:

Given T as a set of tasks, P as performance metric, E as type of experience, and S as a set of learning stimuli, a computing system perpetually learns with regard to (T, P, E, S) if the system consistently and continuously improves its performance P at T, following E and S over time.

**Inconsistencies as learning stimuli:**

Identifying learning stimuli explicitly helps develop stimulus-specific learning methods or strategies. Learning in episodic stimulus specific method is more productive to environment than that of continuous learning process.

Learning process is generally triggered by events, stimuli, desires, goals or objectives. Inconsistencies, contradictions, conflicts, peculiarities, anomalies, outliers, or surprises can be used as stimuli to learning because they often signify; the boundaries or gaps of an agent's know-how: what the agent possesses in its knowledge base cannot properly and adequately handle encountered inconsistent circumstances.

The inconsistency-induced learning process is directly geared toward overcoming encountered inconsistencies. As a result, the agent is able to patch or stretch the boundaries of its knowledge with regard to the inconsistent circumstances such that it knows how to handle the similar situation when it arises next time.

Mismatching inconsistency:

Definition:

If an agent's decision or reasoning process gives rise to a mismatching compound predicate and its defining logical expression, then there exists mismatching inconsistency.

Consider a compound predicate (L) that is fully defined through a logical expression ($L_1$) which is other predicates.

For instance, a mobile agent can be defined as one that can be executed on two different hosts:

*Mobile agent(x) = [Executing(x, host1) ∧ Executing(x, host2) ∧ host1 ≠ host2]*

Here the compound predicate *Mobile agent* is fully defined by the logical expression

*(L1) = [Executing(x, host1) ∧ Executing(x, host2) ∧ host1 ≠ host2]*

Given a compound predicate (L) and it's fully defining logical expression ($L_1$),

If we have either of the following circumstances:

- An object *x* satisfies the compound predicate, but does not satisfy one of its defining logical conjuncts.
  *For instance, Mobile agent (x) ∧ ¬Executing (x, host1)*

- An object *x* does not satisfy the compound predicate, but does satisfy the fully defining logical expression.

  For example, the following is no longer valid:

*¬Mobile agent(x) ∧ [Executing(x, host1) ∧ Executing(x, host2) ∧ host1 ≠ host2]*

Then we say that the compound predicate and its defining logical expression are *mismatching*.

Asymmetric inconsistency:

Definition:

If an agent's decision or reasoning process creates asymmetric literals, then there exists asymmetric inconsistency.

For a predicate *P* representing a symmetric relation, we have

$$\forall x \forall y [(P(x, y) \supset P(y, x)) \wedge (P(y, x) \supset P(x, y))]$$

Let *L1 = P (x, y)* and *L2 = P(y, x), L1 and L2* are referred to as *symmetric*.

For instance, given that L1=Connected (agent1, agent2) and L2=Connected (agent2, agent1)
We say that L1 and L2 are *symmetric.*

If *L1 and L2* are assertions about a *symmetric relation*,
But L1 and L2 are no longer *symmetric*, we say that *L1 and L2* are *asymmetric*.

For instance, when L1 =Connected (agent1; agent2) and L2 = ¬Connected (agent2; agent1), then L1 and L2 are *asymmetric* literals.

**Framework:**

Since the inconsistent phenomena are primarily considered as learning stimuli, we refer to the framework as inconsistency-induced learning, or i2Learning for short.

For developing intelligent agent systems that can automatically and incrementally improve their problem-solving performance over time through perpetual learning; we need to have a framework that

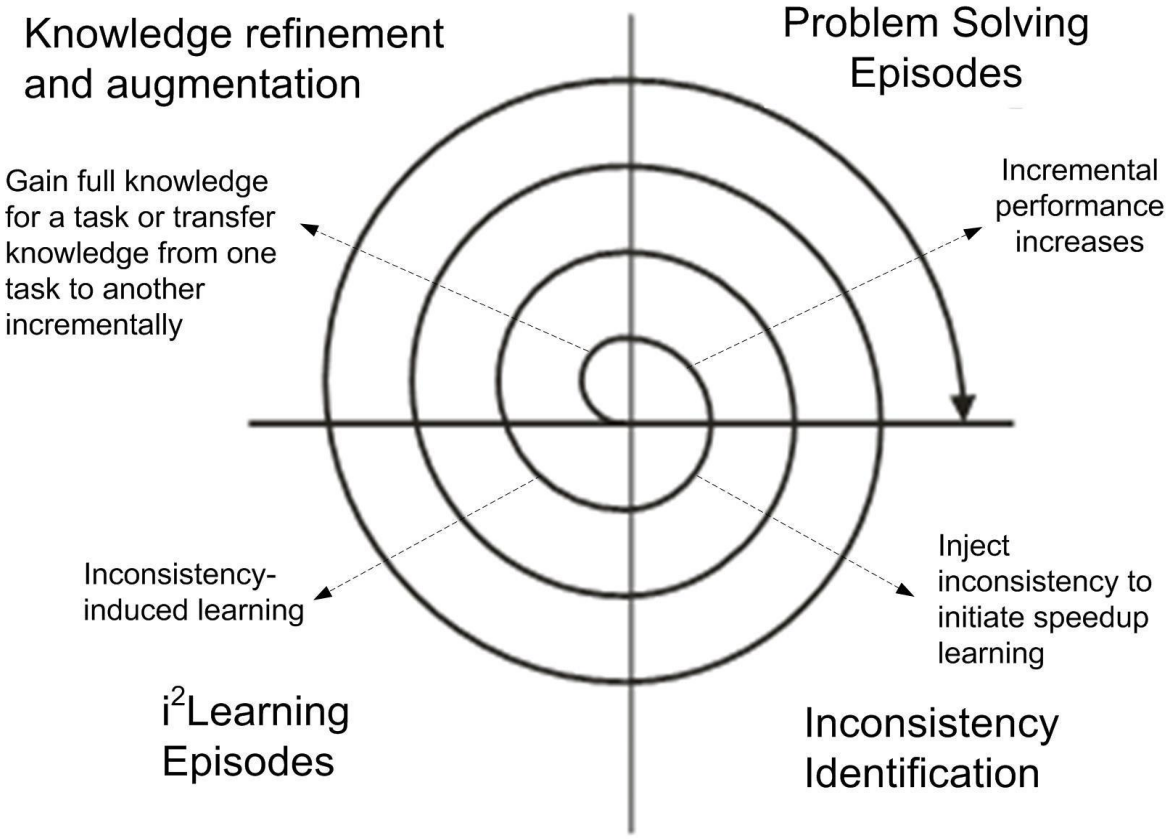
(1)Allows an agent to be engaged in an alternating sequence of problem-solving episodes and learning episodes;

(2)Recognizes a set of learning stimuli S to initiate learning episodes;

(3)Accommodates stimulus-specific learning algorithms and strategies;

(4)Supports a feedback loop to refine or augment the agent's existing knowledge at the end of a learning episode.

Spiral model for perpetual learning:




In spiral model of episodic and perpetual learning:

(1) We define agent's life cycles as an alternating sequence of problem-solving episodes and learning episodes and allowing an agent to stay in its problem-solving episodes until some learning trigger arises

(2)The two distinct types of episodes are defined: problem-solving episodes and inconsistency induced learning episodes for intelligent agent system.

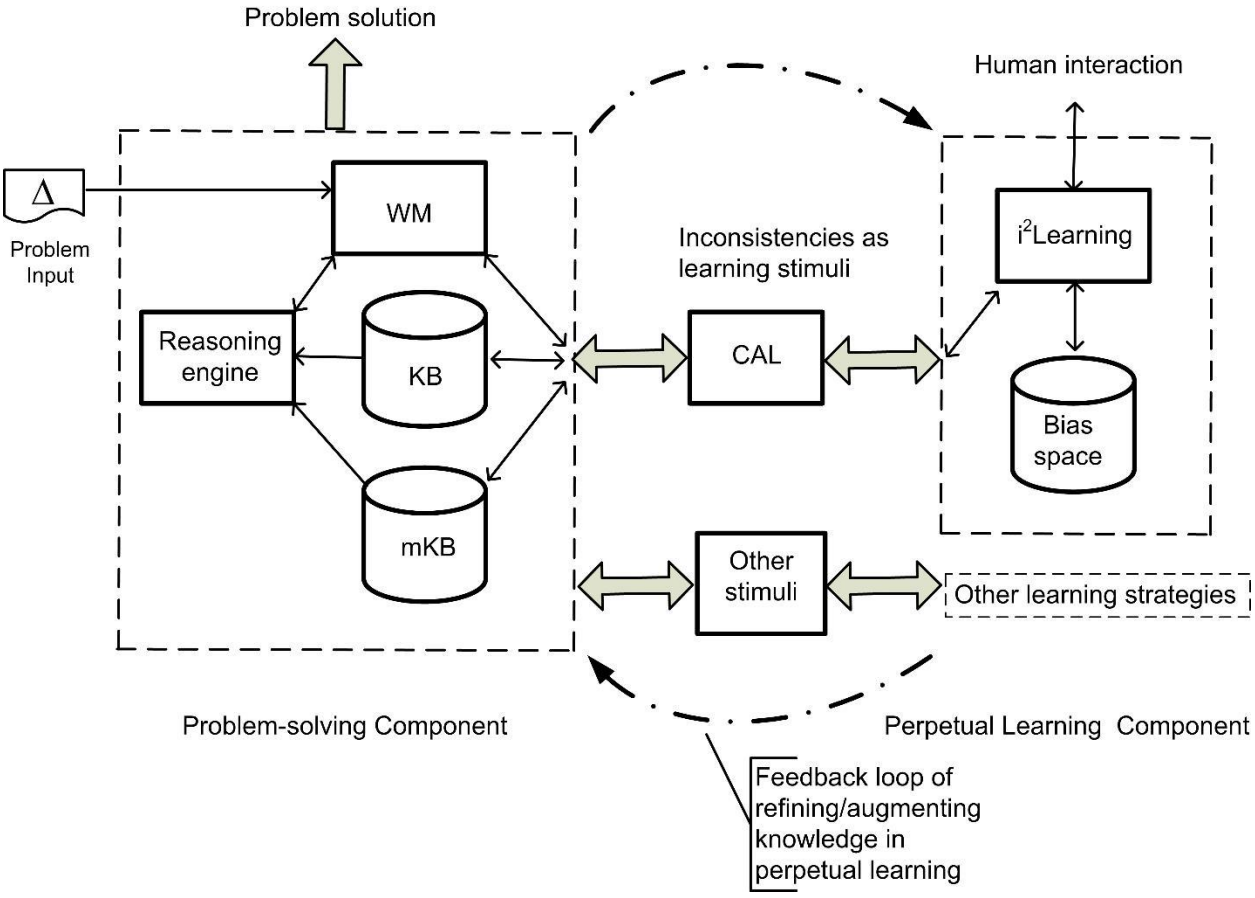
An agent will remain in problem-solving episodes until it encounters some type of inconsistency or conflicting circumstance, at which time the encountered inconsistency serves as a learning trigger for the agent's next learning episode.

(3) It accommodates the inconsistency-specific heuristics to handle various types of encountered inconsistencies

(5) Allows agent's knowledge to be continuously refined or augmented for incremental performance improvement. Learning is embodied in the process of overcoming inconsistencies. Each of such learning episodes results in an agent's existing knowledge being revised toward improved performance.

i2Learning framework:

The inconsistency induced learning process overcomes encountered inconsistencies by patching or stretching the boundaries of its knowledge.




Consist of a problem-solving component and a perpetual learning component.

(1) Knowledge base (KB) for the agent's persistent knowledge and beliefs, and ontological constraints, assumptions, and defaults for its problem domain

(2) Meta knowledge base (mKB), knowledge on how to apply domain specific knowledge in KB during problem solving

(3)Working memory (WM) that holds problem specific facts from input, and facts deduced with activated beliefs from KB

(4)Reasoning mechanism for the problem solving process

(5)CAL (**C**oordinator for **A**pplying episodes and **L**earning episodes) for

i. Detecting inconsistencies, initiating learning episodes
ii. Switching between problem-solving episodes and learning episodes
iii. Refining, augmenting agent's knowledge at the end of learning episodes

When a conflicting situation arises in WM during problem-solving episode, the agent's CAL detects it, suspends the current problem-solving session, initiates the next learning episode, and waits for the result from the i2Learning module.

(6)Bias space of candidate learns biases that certain i2Learning algorithms rely on

(7) The i2Learning module of various inconsistency-induced learning algorithms that carry out the actual learning process to overcome encountered inconsistencies and to refine or augment KB, mKB, or WM (or any combination of the three). The outcome of the learning process is an incrementally improved problem-solving performance.

The i2Learning module carries out the learning process by recognizing the category and a morphology of inconsistency in the conflicting circumstance, and selecting the appropriate inconsistency-specific learning algorithm to overcome the inconsistency that in turn results in its knowledge (KB, mKB, WM) being refined, revised or augmented through CAL.

## i² learning heuristics and algorithms:

Mismatching inconsistency:

Mismatching inconsistency refers to mismatched derivation of a compound predicate (L) and it's fully defining logical expression ($L_1$), denoted as $L \neq L_1$.

To facilitate learning through overcoming mismatching inconsistencies, we define;

- $\mathfrak{I}$ be a set containing conflicting circumstances or contradicting pieces of evidence expressed as pairs of mismatching literals
- $\aleph$ denote the set of heads (conclusions) of the rules that explain the inconsistency.
- $\wp$ contains the set of predicates that appear in WM (working memory) and that will be utilized during the learning bursts.
- $\Re$ be the set of refined rules produced at the end of a learning burst. The learning burst will refine the rule and the set $\Re$ of revised rule(s) will be the output of the algorithm.

Step 1: Once KB∪WM yields some conflicting literals, CAL detects the inconsistency and places mismatching literals in $\mathfrak{I}$.
Step 2: It then generates $\wp$ based on the content of WM and produces $\aleph$ through dependence analysis with regard to predicate(s) in $\mathfrak{I}$.
Step 3: Afterwards, CAL passes $\mathfrak{I}$, $\wp$, $\aleph$ to the algorithm below as input. The algorithm, adopted from uses Foil_Gain as the heuristics in refining rule(s) that have their heads in $\aleph$.
Step 4: The set $\Re$ of revised rule(s) will be the output of the algorithm.

Algorithm for i2Learning(DEC, mim)

Input : $\mathfrak{I}$, $\wp$, $\aleph$
Output: $\Re$

$K_p$: set of ground predicated from KB in $\wp$
$W_p$: set of ground predicated from WM in $\wp$

$\Re = \emptyset$;

$\wp$={ {Kp}, {Wp} }; //Conflicting Predicates

while ($Wp \neq \emptyset$) do { //Predicates from WM

$R_p$ = argmax{Foil_Gain(Wp, $\aleph$); //Chooses predicate to add

while(EXIST($R_p$ in $K_p$))

$\Re = \aleph \wedge R_p$;

$\Re = \Re \cup \{R_p\}$

}

return($\Re$);

Example:

Let $KB_0$ contains knowledge;

Purchase(x) ←up_trade(x)

¬ Purchase(x) ←¬up_trade(x)

up_trade(x) ←Good_standing_Share(x)

Good_standing_Share(s1)

Good_standing_Share (s2)

And $WM_0$ has following facts;

Purchase (s1)

¬ Purchase (s2)

¬Daily_drop (s1, 15%)

CAL detects that there exists the following mismatching inconsistency in KB0∪WM0

KB0∪WM0 ⊢ {Purchase (s2), ¬ Purchase (s2)}

The mismatching inconsistency causes CAL to initiate an episode of learning. The input passed to the learning module from CAL consists of the following:

$\Im$ = {Purchase (s2), ¬ Purchase (s2)}
$\wp$ = {up_trade, ¬Daily_drop}
$\aleph$ = {Purchase}

The algorithm will use Foil_Gain to select a predicate in $\wp$ to be added to the condition part of the rule with "Purchase" ($\in\aleph$) as its head so as to further refine the rule. The learning burst will refine the "Purchase" rule into the following:

$\Re$ = {Purchase (x) ← up_trade (x) ∧ ¬Daily_drop (x, 15%)}

Hence KB1 now contains the following:

Purchase (x) ← up_trade (x) ∧ ¬Daily_drop (x, 15%)

¬ Purchase(x) ←¬up_trade(x)

up_trade(x) ←Good_standing_Share(x)

Good_standing_Share(s1)

Good_standing_Share (s2)

Mismatching inconsistency refers to mismatched derivation of a compound predicate (L) and it's fully defining logical expression ($L_1$), denoted as $L \neq L_1$.

Asymmetric inconsistency:

Let's consider, *L1 = Purchase (share1, share2)* and *L2 = Purchase (Share2, share1), L1 and L2* are referred to as *symmetric*, If *L1 and L2* are assertions about a *symmetric relation*.

But if *L1 and L2* are no longer *symmetric*, we say that *L1 and L2* are *asymmetric*. For instance, when *L1 = Purchase (share1, share2)* and *L2 = Purchase ¬ (Share2, share1)*, then L1 and L2 are *asymmetric* literals.

(This scenario may arrive due to difference in the 'per day drop' and 'up/down trade', etc. facts about both the share1 and share2 which are changing periodically)

To establish learning through overcoming asymmetric inconsistencies;

We define two literals *P* and *Q,*

- 'η' a supporting function such that if there exist a precedence relationship between P and Q, we use $P \succ Q$ to denote that P precedes Q with supporting function 'η'.
- 'π' *weightage* such that, π (*P*) and π (*Q*) to denote the *weight* information (expressed in terms of priority, importance, or significance, etc.) for *P* and *Q*, respectively.
- 'φ' *constraint violation* such that, φ (*P*) (or φ (*Q*)) to denote the fact that the presence of *P* (or *Q*) violates some domain constraint.

Given an instance of asymmetric inconsistency;

$\Im$ = {P (a, b), ¬Q (b, a)} ∨ {¬P (a, b), Q (b, a)} Where, P and Q are asymmetric, or

$\Im$ = {P (a, b), Q (b, a)}

Where, P and Q are symmetric,

After generalizing the literals in $\Im$ by replacing ground terms with variables. (e.g., from P (a, b), Q (b, a) to P (x, y), Q (y, x))

To form $\mathbb{C}$ = {P (x, y), Q (y, x)} (or $\mathbb{C}$ = {P (x, y), ¬ Q (y, x)} or $\mathbb{C}$ = {¬ P

(x, y), Q (y, x)}),

Algorithm: for overcoming asymmetric Inconsistencies

Input: $\Im$ = {P, Q} where P and Q are asymmetric thus,
$\Im$ = {P, ¬Q} ∨ {¬P, Q}
KB, mKB, WM;

Output: refined KB; refined mKB; refined WM;

\\ For supporting function 'η' along with constraint violation check φ

if [KB⊢η(P) ∧ KB⊢η(Q)] then {

if (φ(η(Q))∧¬φ(η(P))) then

{ mKB = mKB ∪ ℂ: η(P)/ η(Q); KB = KB ∪ {¬Q}; break}

elseif (φ(η(P))∧¬φ(η(Q))) then

{ mKB = mKB ∪ ℂ: η(Q)/ η(P); KB = KB ∪ {¬P}; break }

elseif (φ(η(P))∧φ(η(Q))) then

{ break to human handling};

\\ For weightage function 'π' along with constraint violation check φ

if (π(η(P)) > π(η(Q))) then

if (φ(π (Q))∧¬φ(π (P)))

{ mKB = mKB ∪ ℂ: η(P)/ η(Q); KB = KB ∪ {¬Q}; break}

elseif (π(η(Q)) > π(η(P))) then

if (φ(π (P))∧¬φ(π (Q)))

{ mKB = mKB ∪ ℂ: η(Q)/ η(P); KB = KB ∪ {¬P}; break};

elseif (φ(π (P))∧φ(π (Q))) then

{ break to human handling};

We use the following notation to denote that the support function above the line is preferred over the support function below it with regard to the circumstance as specified by ℂ:

1. P and Q are symmetric :

$$\{P, Q\} \qquad \frac{\mathbb{C}: \eta(P)}{\eta(Q)}, \quad \frac{\mathbb{C}: \eta(Q)}{\eta(P)}$$

2. P and Q are asymmetric :

$\{P, \neg Q\}$ $\quad \frac{\mathbb{C}: \eta(P)}{\eta(\neg Q)}, \quad \frac{\mathbb{C}:\eta(\neg Q)}{\eta(P)}$

$\{\neg P, Q\}$ $\quad \frac{\mathbb{C}:\eta(\neg P)}{\eta(Q)}, \quad \frac{\mathbb{C}: \eta(Q)}{\eta(\neg P)}$

Thus, given the $\Im$ (interpretation) containing asymmetric inconsistency. Circumstances to choose support function over another depend on;

- Subclass – superclass generality
- Most recent function
- Higher priority
- Violation of domain constraint
- Derivability from KB

At the end of each of these inconsistencies specific learning heuristic the knowledge base of agent system is refined/ augmented by feedback loop from the i2learning model that improves systems performance by perpetually learning.

**Conclusion:**

Studied the definitions of machine learning and perpetual learning with accordance to the first class entities, and the basic concepts regarding perpetual learning agent system like need of perpetual learning and how machine learning can accomplish that.

Discussed the role of inconsistencies in perpetual learning and how they help triggering learning stimuli in the $i^2$learing framework.

Described the basic functionality requirements in the framework of perpetual learning agent system and working of the $i^2$learing framework along with the spiral model.

The $i^2$Learning heuristics and algorithm to handle mismatching and asymmetric inconsistencies is suggested and demonstrated working of the algorithms for the real world problem in automated algorithmic trading system/ robot where the asymmetric and mismatching inconsistencies may occur.

In future work these algorithms can be refined and modified for more efficiency and precise results. Other real world application can be identified for utilization of above suggested heuristics and algorithms for mismatching and asymmetric inconsistency.

## References

[1]T. Mitchell, The discipline of machine learning, McGraw-Hill

[2]D. Zhang, Perpetual learning through Overcoming Inconsistencies, ICTAI 2013

[3]D. Zhang, Learning through Overcoming Inconsistencies

[4]Du Zhang and M. Lu, Inconsistency-induced learning for perpetual learners, *International Journal of Software Science and Computational Intelligence*, 2011

[5]D. Zhang and M. Lu, Learning through overcoming inheritance inconsistencies, *Proc. of 13th IEEE International Conference on Information Reuse and Integration*, Las Vegas, NV, August 2012.

[6]D. Zhang, Learning through overcoming incompatible and antisubsumption inconsistencies, *Proc. of the 12th IEEE, International Conference on Cognitive Informatics*, New York City, NY, July 2013.

[7]D. Zhang, Perpetual learning through Overcoming Inconsistencies, ICTAI 2013