

---

18.

Abstraction, Reformulation & Approximation

---

# Outline

---

- Introduction
  - Abstraction
  - Reformulation
  - Approximation
  - Summary
-

# Introduction

---

- Since the early days of AI, many have argued that abstraction, reformulation, and approximation (ARA) are central to human reasoning and problem solving and to the ability of computer systems to reason effectively in complex domains
  - The primary use of ARA techniques has been to overcome computational intractability by decreasing the combinatorial costs associated with searching large spaces. In addition, ARA techniques are useful for knowledge acquisition and explanation generation in complex domains
-

# Introduction

---

- The distinction between the terms abstraction, reformulation and approximation is not sharp
  - We will use the following working definitions for this lecture
    - abstraction- ignoring some details
    - reformulation- changing the ontology
    - approximation – concepts that defy complete definitions
  
  - We will begin by considering examples of ARA that have come up in the course so far
  - We will consider more examples from the literature later on
-

## Example from HW2

---

- The **human heart** is a muscular [organ](#) that provides a continuous [blood circulation](#) through the [cardiac cycle](#) and is one of the most vital organs in the [human body](#).<sup>[1]</sup> The heart is divided into four main [chambers](#): the two upper chambers are called the left and right [atria](#) and two lower chambers are called the right and left [ventricles](#). There is a thick wall of muscle separating the right side and the left side of the heart called the [septum](#). Normally with each beat the right ventricle pumps the same amount of blood into the [lungs](#) that the left ventricle pumps out into the body. Physicians commonly refer to the right atrium and right ventricle together as the **right heart** and to the left atrium and ventricle as the **left heart**.<sup>[2]</sup>
-

## Example from HW3

---

California has a “Basic Speed Law.” This law means that you may never drive faster than is safe for current conditions. For example, if you are driving 45 mph in a 55 mph speed zone during a dense fog, you could be cited for driving “too fast for conditions.” You may never legally drive faster than the posted speed limit, even if you think it is safe.

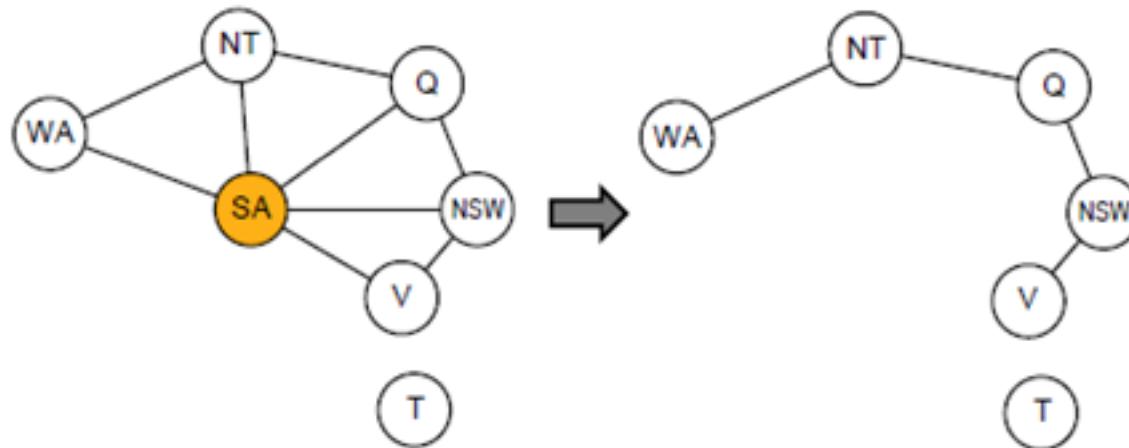
---

# Example of Abstraction

(from lecture on constraint satisfaction problems)

---

Conditioning: instantiate a variable, prune its neighbors' domains



Cutset conditioning: instantiate (in all ways) a set of variables such that the remaining constraint graph is a tree

Cutset size  $c \Rightarrow$  runtime  $O(d^c \cdot (n - c)d^2)$ , very fast for small  $c$

---

# Example of Reformulation (From lecture on Classical Planning)

---

- From Situation Calculus

Given a formula  $Goal(s)$ , find a sequence of actions  $a$  such that

$$KB \models Goal(do(a, S_0)) \wedge Legal(do(a, S_0))$$

where  $do(\langle a_1, \dots, a_n \rangle, S_0)$  is an abbreviation for

$$do(a_n, do(a_{n-1}, \dots, do(a_2, do(a_1, S_0)) \dots))$$

and where  $Legal(\langle a_1, \dots, a_n \rangle, S_0)$  is an abbreviation for

$$Poss(a_1, S_0) \wedge Poss(a_2, do(a_1, S_0)) \wedge \dots \wedge Poss(a_n, do(\langle a_1, \dots, a_{n-1} \rangle, S_0))$$

- to STRIPS

GoThru( $d, r_1, r_2$ ):

precondition: InRoom(robot,  $r_1$ ), Connects( $d, r_1, r_2$ )

delete list: InRoom(robot,  $r_1$ )

add list: InRoom(robot,  $r_2$ )

Goal: [ Box( $x$ )  $\wedge$  InRoom( $x, room_1$ ) ]

---



## Example of Approximation (from the lecture on defaults)

---

- Introduce an abnormality predicate  $Ab$  to talk about the exceptional or abnormal cases where the default should not apply

$$\forall x[\text{Bird}(x) \wedge \neg Ab(x) \supset \text{Flies}(x)]$$

- Here, the  $Ab$  predicate is an approximate predicate where the notion of abnormality is not precisely defined
-

# Abstraction

---

- We informally define abstraction as (Giunchiglia, Walsh 1992):
    1. *The process of mapping a representation of a problem, called the "ground" representation, onto a new representation, called the "abstract" representation, which:*
    2. *helps deal with the problem in the original search space by preserving certain desirable properties and*
    3. *is simpler to handle as it is constructed from the ground representation by "throwing away details".*
-

# Abstraction

---

- A *formal system* is a triple  $\langle L, D, W \rangle$  where  $W$  is a language,  $D$  is deductive machinery (set of inference rules), and  $W$  is a set of axioms in  $L$
- $\text{Th}(\langle L, D, W \rangle)$  is minimal set of formulas in  $L$  containing  $W$  and closed under  $D$
- $F$  is set of formal systems
- An abstraction, written  $f: F1 \rightarrow F2$  is a pair of formal systems  $(F1, F2)$  with languages  $L1$  and  $L2$  respectively and an effective total function  $f_A: L1 \rightarrow L2$

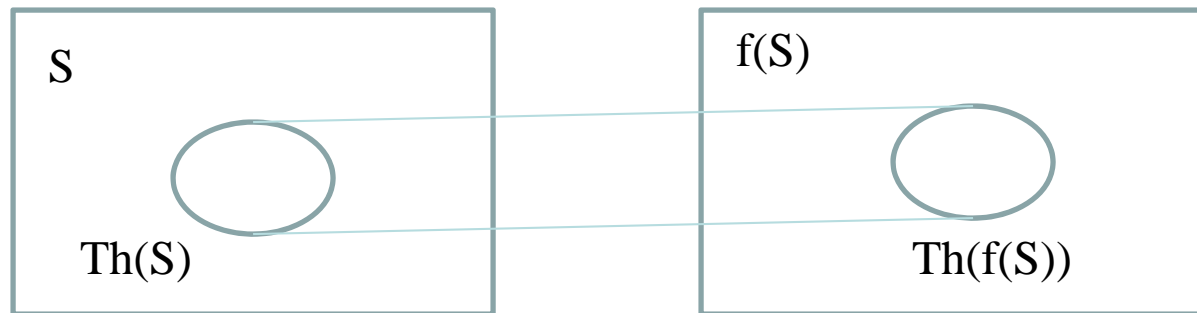
# Classification of Abstractions

---

- Driven by whether an abstraction preserves provability
    - TC Abstraction
      - T stands for theorem, C for constant
    - TD Abstraction
      - D stands for decreasing
    - TI abstraction
      - I stands for increasing
-

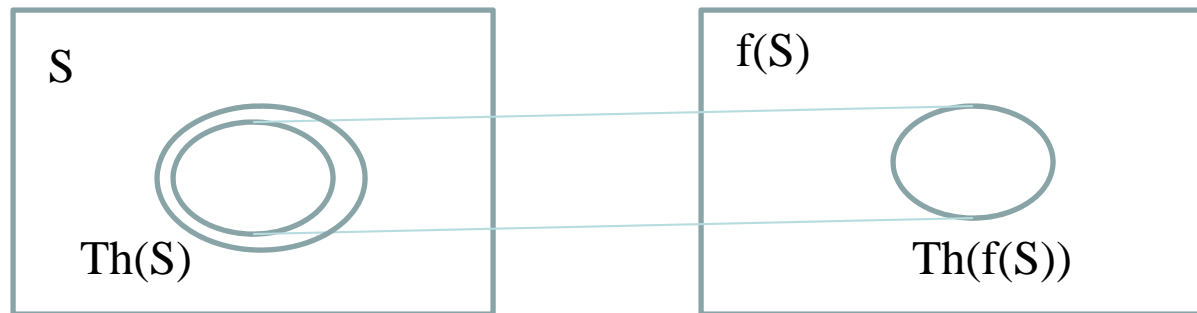
# TC Abstraction

- 
- $f$  is *TC abstraction* iff, for all formulas  $p$ ,  $p \in \text{Th}(S)$  iff  $f(p) \in \text{Th}(f(S))$



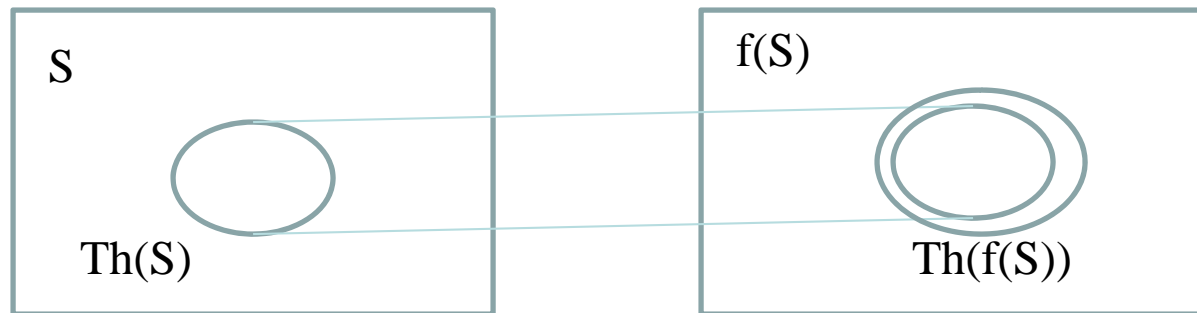
# TD Abstraction

- 
- $f$  is *TD abstraction* iff, for all formulas  $p$ , if  $f(p)$  in  $\text{Th}(f(S))$  then  $p$  in  $\text{Th}(S)$



# TI Abstraction

- 
- $f$  is *TI abstraction* iff, for all formulas  $p$ , if  $p \in \text{Th}(S)$ , then  $f(p) \in \text{Th}(f(S))$



## Example Abstraction ABSTRIPS

---

- In a STRIPS formulation of the planning problem
    - A state is specified using a set of ground atomic wffs
    - Actions are represented as operators with pre-conditions, add lists and delete lists
  - ABSTRIPS was one of the first uses of abstraction [Sacerdoti, 1973]
    - Each pre-condition has associated with it a criticality
-



## Example Abstraction ABSTRIPS

---

- Consider the operator Turn on the Lamp with the following pre-conditions:  
     $\text{Type}(L, \text{Lamp}) \wedge \text{InRoom}(\text{Me}, \text{Rx}) \wedge \text{InRoom}(L, \text{Rx}) \wedge$   
     $\text{Plugged-In}(L) \wedge \text{NextTo}(\text{Me}, L)$
  - We can associate criticality with each pre-condition  
    [4]  $\text{Type}(L, \text{Lamp}) \wedge$  [3]  $\text{InRoom}(\text{Me}, \text{Rx}) \wedge$  [3]  $\text{InRoom}(L, \text{Rx}) \wedge$   
    [2]  $\text{Plugged-In}(L) \wedge$  [1]  $\text{NextTo}(\text{Me}, L)$
  - Each level of criticality defines an abstract search space. For example, the search space for criticality [4] would have only the following pre-condition  
     $\text{Type}(L, \text{Lamp})$
  - ABSTRIPS is a TI Abstraction
-

# Reformulation

---

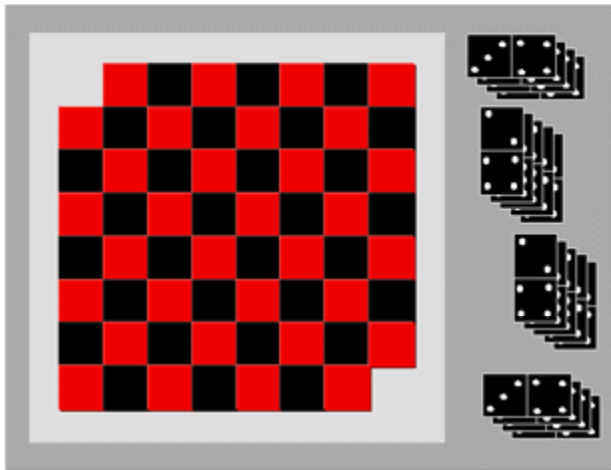
- When we conceptualize a problem, we identify the objects, functions and relations needed to state the problem
  - Reformulation is about changing the objects, functions and relations needed to formulate the problem
-

# Reformulation Example

---

## Mutilated Checkerboard Problem

Suppose that a black square has been cut from each corner of a Checkerboard so that it now contains 62 squares. You are given a set of 31 dominoes, where each domino covers exactly two squares of the checkerboard. Can you specify an arrangement of the dominoes on this checkerboard such that the dominoes exactly cover this mutilated checkerboard?



Formulation 1:  
62 distinct objects

Formulation 2:  
30 objects of one color  
32 objects of another

---

## Reformulation Definitions

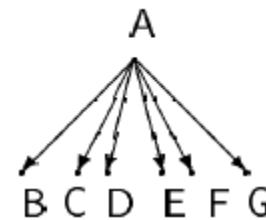
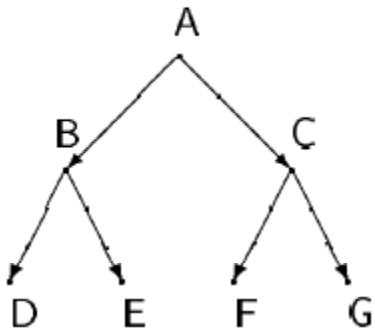
---

- *Conceptualization*: is a triple  $(O, F, R)$  where  $O$  is a set of objects,  $F$  is called the functional basis set and is a subset of functions from  $O^n$  to  $O$ , and  $R$ , called the relational basis set, is the subset of relations on  $O^m$ , for  $n, m$  in the set of natural numbers
- An encoding  $E$  of a conceptualization  $C$  is a set of sentences in canonical language  $L_C$  such that  $C$  is one of the models of  $E$
- A conceptualization  $C_2$  is a re-conceptualization of  $C_1$  with respect to some background conceptualization  $D$ , if the elements of  $C_2$  are definable from  $C_1$  in  $D$ .
- $C_2$  is a correct re-conceptualization of  $C_1$  with respect to  $D$  and the set of goal relations  $G$ , if  $G$  is definable in both  $C_2$  and  $C_1$

## Reformulation Example

---

- Objects: the set  $P$  of People  $\{A, B, C, D, E, F, G\}$
  - Functions: the function Father from  $P$  to  $P$
  - Relations: the relation ancestor which is a subset of  $P^2$
- Objects: the set  $P$  of People  $\{A, B, C, D, E, F, G\}$
  - Relations: the relation FoundingFather which is a subset of  $P^2$



Same Family relation is preserved across the two conceptualizations

---

# Types of Reformulations

---

- Syntactic/logical
  - Models of the reformulated theory are the same as the models of the original theory
- Semantic
  - Models of the reformulated theory can be different from the models of the original theory
  
- We will consider several examples of these reformulations

# Syntactic Reformulation

---

- Problem solver

- Prolog

- Database

- $g(X, Y) : -p(X), q(X, Y)$

- $g(X, Y) : -p(X), r(X, Y)$

- Reformulation

- $g(X, Y) : -p(X), q(X, Y) \mid r(X, Y)$

- Semantic Equivalence

- Same set of models

# Semantic Reformulation

---

- Let  $M(T)$  be the models of theory  $T$
- Let  $r(T)$  be the reformulated theory
- Semantic equivalence
  - $M(T) = M(r(T))$
- In some cases, semantic equivalence may not be preserved



# Ontological Reification

---

- Reification is the process of adding new objects to a schema and new relations on those objects to represent information previously expressed entirely with relations
    - See the chapter on Master Schema Management for more details
      - <http://logic.stanford.edu/dataintegration/chapters/chap05.html>
-

# Ontological Reformulation

---

red(a)

green(a)

blue(c)

color(a, red)

color(b, green)

color(c, blue)

property(color, a, red)

property(color, b, green)

property(color, c, blue)

---

# Ontological Reformulation

---

- Reification can sometimes simplify the encoding of transformation rules and decreases the number of conjuncts in a query
  - Consider defining a relation  $r$  consisting of all pairs of objects satisfying a relation  $p$  such that the objects in each pair have different colors
  - Query in Un-reified schema
    - $r(X,Y) :- p(X,Y) \& \text{red}(X) \& \text{green}(Y)$
    - $r(X,Y) :- p(X,Y) \& \text{red}(X) \& \text{blue}(Y)$
    - $r(X,Y) :- p(X,Y) \& \text{green}(X) \& \text{red}(Y)$
    - $r(X,Y) :- p(X,Y) \& \text{green}(X) \& \text{blue}(Y)$
    - $r(X,Y) :- p(X,Y) \& \text{blue}(X) \& \text{red}(Y)$
    - $r(X,Y) :- p(X,Y) \& \text{blue}(X) \& \text{green}(Y)$
  - Query in reified schema
    - $r(X,Y) :- p(X,Y) \& \text{color}(X,U) \& \text{color}(Y,V) \& U \neq V$
-

# Conceptual Reformulation

---

- Problem solver

- Prolog

- Database

$Car(x) \leq JapaneseCar(x)$        $Car(x) \leq EuropeanCar(x)$   
 $JapaneseCar(x) \leq Toyota(x)$      $EuropeanCar(x) \leq BMW(x)$

- Reformulation

$Car(x) \leq ForeignCar(x)$   
 $ForeignCar(x) \leq Toyota(x)$        $ForeignCar(x) \leq BMW(x)$

- Semantic Equivalence

- Gives equivalent answers to the query  $Car(x)$
  - But, if the KB contained:

$Fast(x) \leq EuropeanCar(x)$      $Reliable(x) \leq JapaneseCar(x)$

With these axioms one can infer incorrect conclusions such as European cars are reliable

---

# Functions vs Relations

---

- When we use functions in our ontology
    - We are guaranteed that the value exists
    - The uniqueness of the value is implied
      - If we use `mumof` to represent the mother relation, we are ensured that every individual has a unique mother
      - If we use mother relation, e.g., `mother(X,Y)`, there are no such guarantees
  - For a certain class of theories, we can reformulate them so that each function is replaced by a relation and cardinality constraints
    - A theory containing no functions is decidable
    - No decidable procedure exists for theories containing functions
-

# Approximation

---

- Approximate concepts cannot have “if and only if” definitions and may not even have definite extensions
  - Some approximate concepts can be refined by learning more and some by defining more and some by both, but it isn't possible in general to make them well-defined
  - A sentence involving an approximate concept may have a definite truth value even if the concept is ill-defined.
    - [Based on McCarthy 2000]
-

## Example Approximate Concepts

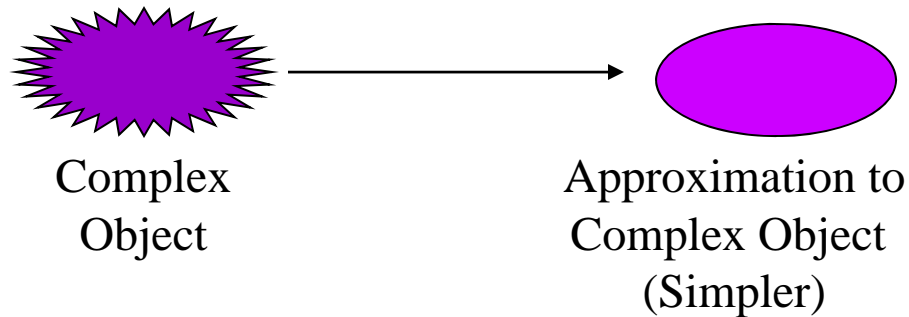
---

- It is definite that Mount Everest was climbed in 1953 even though exactly what rock and ice is included in that mountain is ill-defined
  - It harms a mosquito to be swatted, although we haven't a sharp notion of what it means to harm a mosquito
  - Whatif( $x,p$ ), which denotes what  $x$  would be like if  $p$  were true
-

# Example Approximate Concepts

---

- Objects that represent epistemologically richer objects
  - **Example:** Blocksworld only characterizes blocks by what they rest on ( $On(x,y,s)$ ), not, weight, size, etc.
  - Ignored properties are usually irrelevant, so that representation is simplified.





## Example Approximate Concepts

---

- Conception of objects which have no basis in reality.
  - **Example:** “middle-aged”, red, heaps, “U.S.Wants”.
  - A category constructed to facilitate reasoning.
  - Since definitions are incomplete, fall into paradoxes.
  - Often compositional.

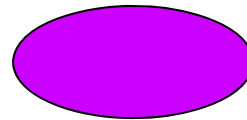
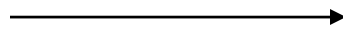


## Example Approximate Concepts

---

- Objects in counterfactual sentences:
  - **Example:** “*car*” in “*If another car had come over the hill, ..., you would have had a head-on collision.*”
  - Similar to Type II objects in that they do not correspond to some real object.
  - Their properties are only those ascribed to them by the counterfactual.
  - (“*car*” above is not a Buick, Mercedes, etc.)

?



Type III  
Approximate  
Object

# Propositional Approximate Theories

---

- Reality
  - Given by a set of propositional variables  $r_1, \dots, r_n$
  - Reality may not be directly observable and  $n$  may be very large
- Observations
  - Let the values of the propositions  $o_1, \dots, o_k$  be observable. They are functions of reality given by
    - $q_j = Q_j(r_1, \dots, r_n)$ ,  $j < k$
  - where  $k$  is a modest number
- An approximate theory AT is given by functions  $Q'_j(o_1, \dots, o_k)$ ,  $j < k$ 
  - It gives us what we want to know in terms of the observations
- If we are lucky, the  $Q'$  functions are the same as  $Q$  functions

$$Lucky(r_1, \dots, r_n) \rightarrow q_i = Q'_i(o_1, \dots, o_k)$$

for  $i = 1, \dots, l$ , i.e.

---

$$Lucky(r_1, \dots, r_n) \rightarrow [Q_i(r_1, \dots, r_n) \equiv Q'_i(O_1(r_1, \dots, r_n), \dots, O_k(r_1, \dots, r_n))].$$

---


# Interesting Questions

---

- Much of the research on ARA has focused on computational issues
    - Significant focus on CSP, Planning, Search problems
  - How can ARA techniques be applied to ontological / conceptual modeling?
    - Describing concepts at different levels of details
      - <http://learn.genetics.utah.edu/content/begin/cells/scale/>
    - Automated reformulation
      - How can we choose the appropriate ontology in response to a question stated in English?
    - Theory of approximate objects
      - What is the relationship between different logical specifications of an approximate objects?
      - How do we relate different approximate theories?
-

# Current Research

- Symposium on Abstraction, Reformulation and Approximation
- <http://logic.stanford.edu/sara2011/>



**SARA 2011**  
**Symposium on Abstraction, Reformulation, and Approximation**  
**Parador de Cardona, 17-18 July 2011**

**Authors**  
[Call for Papers](#)  
[Paper Submission Site](#)

**Attendees**  
[Lodging](#)

**Associated Conferences**  
[SoCS](#)  
[IJCAI-11](#)

**Past SARA Conferences**  
[2009 - Lake Arrowhead](#)  
[2007 - Whistler](#)

**Introduction**

Since the early days of Artificial Intelligence, many have argued that abstraction, reformulation, and approximation (ARA) are central to human common-sense reasoning and problem solving and to the ability of computer systems to reason effectively in complex domains.

The primary use of ARA techniques has been to overcome computational intractability by decreasing the combinatorial costs associated with searching large spaces. In addition, ARA techniques are useful for knowledge acquisition and explanation generation in complex domains.

# Summary

---

- Abstraction, reformulation and approximation concepts are pervasive in
    - Conceptual representation of knowledge
    - Problem solving
  - (Oversimplified) characterization of ARA concepts
    - abstraction- ignoring some details
    - reformulation- changing the ontology
    - Approximation – concepts that defy complete definitions
  - While there is substantial work in using ARA techniques in CSP and planning, little work in knowledge acquisition and explanation generation
-

# Reading

---

- Fausto Giunchiglia, Toby Walsh: "A theory of Abstraction", Journal Artificial Intelligence Volume 57 Issue 2-3, Oct. 1992. [http://dx.doi.org/10.1016/0004-3702\(92\)90021-O](http://dx.doi.org/10.1016/0004-3702(92)90021-O)
  - Subramanian, A Theory of Justified Reformulations  
[http://logic.stanford.edu/publications/subramanian/a\\_theory\\_of\\_justified\\_reformulations.pdf](http://logic.stanford.edu/publications/subramanian/a_theory_of_justified_reformulations.pdf)
  - McCarthy J., Logical Theories with Approximate Objects, In the Proceedings of the International Conference on Knowledge Representation and Reasoning <http://www-formal.stanford.edu/jmc/approximate.html>
-